Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Disseny i implementació d'un joc de geocaching amb Realitat Augmentada

TITULACIÓ: Grau en Enginyeria de Sistemes Aeroespacials

AUTOR: Miguel Martín Gómez

DIRECTOR: Dolors Royo Vallés

DATA: 5 de Juliol de 2019

TÍTOL: Disseny i implementació d'un joc de geocaching amb Realitat Augmentada

AUTOR: Miguel Martín Gómez

DIRECTOR: Dolors Royo Vallés

DATA: 5 de Juliol de 2019

Resum

L'internet, la geolocalització, la Realitat Augmentada, els smartphones... són algunes de les tecnologies més utilitzades i explotades d'avui dia, i quan s'utilitzen de forma conjunta ofereixen un ventall de possibilitats infinit per millorar i facilitar el dia a dia de les persones.

El World Mobile City Project (WMCP) és un projecte col·laboratiu de georeferenciació que es va iniciar al 2012 i que fa ús d'aquestes tecnologies per ensenyar als més joves de diferents instituts de Catalunya i València a moure's per la ciutat. Entre els membres col·laboradors destaca l'equip Lacenet, la Generalitat de Catalunya, els diferents instituts que hi participen, i l'Escola d'Enginyeria de Telecomunicacions i Aeroespacial de Castelldefels.

El WMCP organitza una gimcana duta a terme en les diferents ciutats que hi participen, on els alumnes s'organitzen en grups per realitzar diferents proves en llocs d'interès. Els estudiants han de descarregar-se una aplicació que gestiona les proves de l'activitat donant instruccions i oferint una interfície per interactuar amb l'usuari. Actualment quan l'usuari es baixa l'aplicació ja incorpora tots els elements que conformen les proves i que el docent ha programat prèviament. Per tant la persona encarregada d'organitzar una activitat ha de tenir coneixements informàtics.

Aquest projecte té com a objectiu crear una nova aplicació amb una interfície pels docents, que permeti organitzar activitats des de la pròpia aplicació de forma senzilla i personalitzada, i una pels alumnes que permeti carregar i jugar aquestes partides creades prèviament. És important esmentar que aquesta aplicació no s'ha de considerar com un model definitiu sinó com un model base sobre el que es puguin realitzar modificacions. Per tant hem de pensar en un model flexible que permeti afegir noves funcionalitats sense gaire esforç.

Al llarg d'aquest document s'aniran explicant els mètodes i tecnologies utilitzades, els elements clau per la comprensió d'aquest projecte i els passos que hem seguit per obtenir el resultat final.

TITLE: Design and implementation of an Augmented Reality based geocaching game

AUTHOR: Miguel Martín Gómez

DIRECTOR: Dolors Royo Vallés

DATE: 5 de Juliol de 2019

Overview

Internet, geolocation, augmented reality, smartphones... Are some of today's most used and exploited technologies, and when used together they offer a range of infinite possibilities to improve and facilitate the daily life of people.

The World Mobile City Project (WMCP) is a collaborative geo-referencing project created in 2012 which uses these technologies to teach young people of different Catalan and Valencian institutes to move around the city. Notable members include Lacenet team, Generalitat de Catalunya, the participating institutes, and the Castelldefels School of Telecommunications and Aerospace Engineering.

The WMCP organizes a gymkhana carried out in the different participating cities, where the students are organized in groups to perform different activities in places of interest. Students have to download an application that manages the activities of the gymkhana by giving instructions and offering an interface to interact with the user. Currently, when the user downloads the application, it already incorporates all the elements that make up these activities and that the teacher has previously programmed. Therefore, the staff in charge of organizing an activity must have computer skills.

This project aims to create a new application with an interface for teachers, which allows to organize activities from the application itself in a simple and personalized way, and one other for the students that allows to download and play these games created previously. It is important to mention that this application should not be considered a definitive model but as a base model on which modifications can be made. Therefore, we must think of a flexible model that allows to add new functionalities without much effort.

Throughout this document the used methods and technologies, key elements for the understanding of this project and the steps that we have followed to obtain the final result will be explained.

Agraïments

M’agradaria donar el més sincer agraïment:

Al professor Miguel Valero, el qual em va impulsar i motivar a realitzar aquest projecte.

A la professora Dolors Royo, la qual ha atès tots els meus dubtes al llarg del projecte i sempre s’ha preocupat per buscar un forat i parlar amb mi.

Al senyor Joan Gómez, bon amic de la família, el qual sempre ha estat disposat a oferir-me la seva mà quan ho he necessitat.

A l’EETAC en general, per haver-me brindat la possibilitat de realitzar aquests estudis tan valuosos.

A la meva família, que sempre s’ha preocupat per mi i per la meva educació, per fer-me entendre que les nits d’estudi mentre els meus amics sortien reialment servirien per alguna cosa. Sense ells probablement avui no em trobaria aquí acabant de redactar el treball de fi de grau, el qual marca el final d’una etapa, i el començament d’una altra...

INDEX

INTRODUCCIÓ

Estem a dilluns, dia de classe, ja s'ha acabat el cap de setmana i són les set del matí. Sona l' "alarma"... ¿L' "alarma"? Sí, per que no ens enganyem, molts de nosaltres ja no tenim un despertador d'aquells amb dues potes i un parell de campanes metàl·liques, sinó que escollim entre dues opcions, posposar o finalitzar. I es que els smartphones s'han tornat en una potent eina, que a mesura que ha anat evolucionant ha substituït diferents dispositius en un de sol que ens cap en la butxaca dels pantalons. Podríem dir que portem a sobre un telèfon, una agenda, una enciclopèdia, un calendari, una càmera, una calculadora, un reproductor de musica, un mirall, un disc dur, un televisor, fins i tot una consola de videojocs, i no estaríem enganyant.

Així doncs, per una banda aquests dispositius ens permeten fer-ne ús d'ells per cobrir necessitats bàsiques, com realitzar una cerca ràpida des del navegador quan no ens en sortim amb aquell tema, computar una arrel quadrada des de la calculadora que porta integrada, o activar l'alarma per no arribar demà tard a classe. Per altra banda els smartphones també s'han convertit en una font d'entreteniment que s'ha guanyat sobre tot l'atenció dels més joves amb la sortida de nous videojocs i xarxes socials.

Si barallem totes les cartes que tenim amb una bona visió d'oportunitats ens adonarem que podem utilitzar tot el potencial d'aquestes eines per fer que els més joves aprenguin i es desenvolupin culturalment mentre s'entretenen. Un exemple d'aquesta visió ja té nom i cognoms, i es diu World Mobile City Project (WMCP). WMCP és un projecte col·laboratiu de georeferenciació i tecnologia mòbil, impulsat per l'Equip LaceNet i amb el suport dels centres que hi participen, entre els quals estem nosaltres, l'escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels (UPC-EETAC).

El projecte proposa una activitat de geolocalització a l'aire lliure enfocada per als estudiants dels diferents centres d'ensenyament de Catalunya, amb l'objectiu d'aprendre a localitzar els llocs d'interès de la ciutat mitjançant l'ús d'una aplicació que els estudiants s'han de descarregar. Aquesta aplicació combina tecnologies com internet, la geolocalització i la Realitat Augmentada (a partir d'ara s'abreujarà com RA), i ha estat desenvolupada per l'EETAC, concretament per Dolors Royo i David Lopez Nuevo.

Resumidament, en aquesta activitat els alumnes s'organitzen en grups i cada grup rep uns targetons[1] corresponents a diferents llocs rellevants del municipi. Els estudiants han de buscar aquests llocs caminant per la ciutat i una vegada localitzats, han de treure el corresponent targeton i enfocar-lo des de l'aplicació que s'han descarregat, que incorpora una càmera de RA. Si es troben en un radi

[1] Els targetons són targetes amb un identificador que l'aplicació de realitat augmentada detecta i per tant desbloqueja les activitats corresponents del lloc d'interès. Al llarg del projecte utilitzarem el terme targets en comptes de targetons.

proper al lloc d'interès es desbloquegen una sèrie d'activitats que es manifesten com a caixes de RA i que s'obren quan l'alumne polsa a sobre. Aquestes proves inclouen preguntes de trivial, links a formularis, fins i tot missatges que continguin accions a realitzar pels alumnes.

Actualment, totes les aplicacions de RA es generen a la universitat de forma manual, el codi es modifica i recompila amb les noves dades de cada edició de la Barcelonada. L'objectiu d'aquest projecte és automatitzar la creació de l'aplicació de RA de forma que siguin els propis professors els que generin la seva pròpia versió a partir de la lectura de fitxers de configuració que prèviament han estat compartit a la xarxa.

Aquesta és una primera aproximació a un model d'aplicació flexible i que ha de permetre personalitzar la gimcana entre modalitats de joc i proves, però en un futur es podrien afegir funcionalitats depenent de com es vulgui realitzar l'activitat en el moment.

Aquesta memòria es dividirà en capítols, i cadascun dels capítols tractarà els següents aspectes del projecte:

- **Capítol 1**: Tecnologies utilitzades. En aquest capítol s'expliquen les tecnologies i programari emprats per dur a terme aquest projecte, i que són necessaris per comprendre el mateix.

- **Capítol 2**: World Mobile City Project. En aquest capítol s'explica en què consisteix el World Mobile City Project i quins són els principals objectius i competències.

- **Capítol 3:** Especificacions de l'aplicació. En aquest capítol s'introdueixen les especificacions requerides pel disseny de l'aplicació.

- **Capítol 4:** Implementació de l'aplicació. En aquest capítol s'explica com s'ha anat desenvolupant l'aplicació i quines tècniques s'han utilitzat per resoldre els problemes que han anat sorgint.

- **Capítol 5:** Flexibilitat de l'aplicació. En aquest capítol s'explica de quina manera es podrien afegir noves funcionalitats a l'aplicació en un futur.

- **Capítol 6:** Conclusions. En aquest capítol s'expliquen les conclusions del projecte.

Capítol 1. Tecnologies utilitzades.

1.1. Realitat augmentada.

La RA és una tecnologia que permet superposar elements virtuals sobre una visió de la realitat aportant un valor afegit, la qual cosa ens ajuda a generar experiències que aporten un coneixement rellevant sobre el nostre entorn en temps real.

El terme de RA va sorgir per primera vegada al 1992 quan el científic i investigador Thomas P. Caudell treballava en el desenvolupament d'un dels avions més famosos del món, el Boeing 747. El treballador va observar que els operaris encarregats de l'acoblament de l'aeronau perdien massa temps interpretant les instruccions i va pensar: ¿Que passaria si els operaris tinguessin accés a una pantalla que els anés guiant durant l'acoblament? La seva idea no va triomfar, però en aquell moment va néixer el concepte RA.

Per tal de poder aconseguir la superposició d'elements virtuals en un entorn real, un sistema de RA necessita dels següents elements:

- Hardware[2]:
  - Càmera: Aquest és el dispositiu que captura l'imatge del món real. Pot ser la càmera web d'un ordinador, la d'un smartphone o una càmera que estigui connectada a un dispositiu amb capacitat computacional.
  - Processador: És el principal element del hardware, que mitjançant les ordres d'un programa realitzarà les operacions necessàries per tal de combinar la imatge real amb la informació que s'ha de superposar.
  - Sensors: Es necessita un giroscopi i un acceleròmetre pel sistema. El giroscopi mesura els moviments del dispositiu amb un braç d'accionament i permet que el dispositiu segueixi els gestos i desplaçaments que realitzem sobre ell mesurant la rotació. L'acceleròmetre és capaç de mesurar l'orientació d'un element estacionari respecte la superfície de la Tierra. També es necessita un receptor GPS en el cas que la RA depengui de la posició/coordenades del dispositiu. Tots els smartphones d'avui dia incorporen aquests sensors i receptors entre molts altres més.
  - Pantalla: En la que es pot observar el resultat final dut a terme pel sistema de RA.

[2] Conjunt d'elements físics o materials que conformen una computadora o un sistema informàtic.

- Software[3]: Es necessita un programari que executi tot el procés del sistema de RA dins de l'aplicació que s'estigui utilitzant. En el nostre cas utilitzarem el motor de videojocs multi plataforma Unity junt amb el SDK (equip de desenvolupament de programari) de Vuforia. Cal esmentar que aquest és un element clau del sistema.

- Servidor (Aquest element no és imprescindible en totes les aplicacions de RA): Un servidor juga un paper important en l'emmagatzemament de la base de dades d'imatges virtuals. En funció de la sol·licitud rebuda de l'aplicació RA, imatges virtuals es recuperen del servidor web o del cloud i s'envien a l'aplicació.

- Activador. És un element del món real que el software utilitzarà per reconèixer l'entorn físic i seleccionar la informació virtual que s'hagi d'afegir, com ara una imatge que sigui rica en detalls, amb un bon contrast i sense patrons molt repetits. Aquest activador també pot ser una determinada posició GPS detectada per l'smartphone i registrada com un punt de RA.

A la **Fig 1.1** podem veure un petit esquema on veiem l'ordre en que es relacionen tots aquests elements per aconseguir un sistema RA.

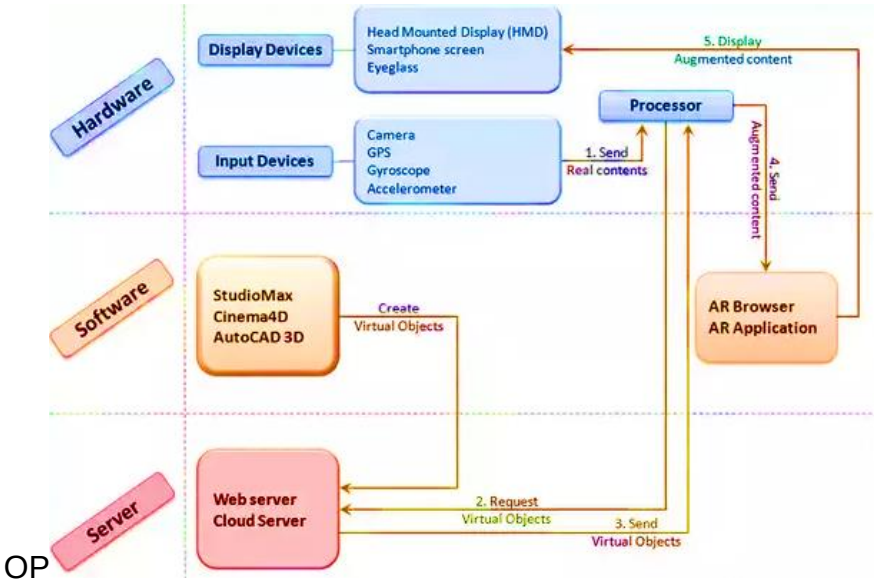

Fig. 1.1. Arquitectura d'un sistema RA.

[3] Conjunt de programes y rutines que permeten a la computadora realitzar determinades funcions.

Podríem dividir la RA en tres grans blocs:

- **Marker-based RA:** Aquest sistema utilitza marcadors i objectes visuals registrats prèviament en el software com a elements activadors. Per tant el dispositiu ha de reconèixer primer aquest identificador i una vegada identificat, l'animació podrà començar d'immediat a superposar els elements virtuals. El sistema de tracking que utilitza el giroscopi i l'acceleròmetre del telèfon s'encarregarà de mostrar els elements en una perspectiva correcta.

- **Markerless RA:** Aquesta tècnica permet l'ús de qualsevol part de l'entorn físic com a objectiu o base per la col·locació d'objectes virtuals superposats. Per aconseguir aquest efecte s'utilitzen tecnologies com l'SLAM (mapatge i localització simultanis). Aquesta tecnologia és capaç de construir un mapa d'entorn desconegut en el que es troba el dispositiu, a la vegada que estima la seva trajectòria i localització al desplaçar-se dins d'aquest entorn.

- **Location-based RA:** Aquesta tècnica vincula el contingut de la RA a una ubicació específica, de manera que quan l'usuari s'apropa a aquesta ubicació apareixen els elements. Es necessita tenir un receptor GPS, una brúixola digital, un mesurador de velocitat i un acceleròmetre activats al smartphone per tal d'anar calculant tota l'estona quina és la ubicació de l'usuari. Alguns usos basats en aquest tipus de RA inclouen el guiatge en la conducció dins d'aplicacions GPS, consulta de serveis propers, guies turístiques virtuals amb informació local, i fins i tot videojocs basats en geolocalització.

En el nostre projecte es combinen la detecció de marcadors amb la geolocalització, de manera que un marcador detectat mostrarà la seva informació virtual a l'usuari depenent de la geolocalització de l'usuari.. Avui dia les aplicacions de la RA són nombroses i cada dia es descobreixen nous àmbits d'ús d'aquesta tecnologia entre els quals destaca l'entreteniment, el turisme, la indústria i l'educació:

- **Turisme:** Aquesta tecnologia permet millorar l'experiència dels visitants a una ciutat o punt d'interès a través de la integració de contingut virtual que aporti informació de la localització en la que es trobin. Fins i tot s'ha arribat a utilitzar per la reconstrucció digital d'elements històrics que ja no estan o que estan deteriorats, com edificis o monuments. La unió entre la RA i el GPS permet també facilitar el trajecte cap al nostre destí a través de directrius visuals precises.

- **Indústria:** El desenvolupament d'aplicacions de RA a la indústria ajuda a millorar la productivitat en els cicles de treball de les empreses. Algunes companyies utilitzen aplicacions que interactuen amb les cadenes de muntatge aportant informació dinàmica als treballadors. Les empreses de reparació de maquinària, també es beneficien d'aquesta tecnologia ja que és capaç d'aportar informació de l'estat i deteriorament de les peces, i fins i tot donar instruccions de reparació.

- **Entreteniment:** Aquest és l'àmbit en el que la RA cobra més pes. La sortida del famós joc Pokémon Go va arrasar en tot el món trencant records amb aproximadament 800 milions de descàrregues. I es que no només la sortida de videojocs ha estat un gran hit. Les xarxes socials com Instagram o Snapchat cada vegada utilitzen més funcionalitats de RA per entretenir als seus usuaris.

- **Educació:** Si tot aquest món virtual ha tingut un impacte en els anteriors àmbits, com no l'anava a tenir en l'educació, terreny en el que pot aportar un gran valor afegit ple de beneficis en quant a aprenentatge-ensenyament.

L'aplicació que desenvoluparem nosaltres està íntimament vinculada a l'educació, l'aprenentatge i l'entreteniment.

1.1.1. Realitat Augmentada a l'educació.

La RA és una tecnologia que ja està present en nombroses aules i les tendències apunten a que continuarà estant-lo. I es que els professors han trobat en aquesta tecnologia l'oportunitat d'explorar l'aprenentatge des d'un punt de vista totalment diferent. "Si afavorim que l'alumnat utilitzi RA a l'aula podem aconseguir que sigui una persona amb millor capacitat d'adaptació i sobretot que potencii la seva creativitat i innovació, habilitats demanades en la societat" va reafirmar Bernat Llopis, professor de formació professional.

El principal objectiu de portar aquesta tecnologia a l'educació es que els estudiants gaudeixin d'un procés d'aprenentatge interactiu i enriquidor, que els motivi i els ajudi a que l'estudi sigui una tasca més lleugera. A més, aquesta eina la porten integrada ja que a casa són molts els que estan acostumats a jugar videojocs de RA, la qual cosa facilita molt la integració a l'aula.

Algunes de les aplicacions i tècniques que s'han implementat a través de la RA es mostren a la següent llista:

- Afegir textos, objectes d'aprenentatge, material educatiu en general o informació complementària als libres de text.

- Aconseguir interactivitat entre el subjecte i l'objecte d'estudi a l'hora de realitzar activitats.

- Enriquir l'entorn educatiu amb elements de realitat augmentada.

- Treballar en el modelatge d'objectes,

- Escoltar i indagar en l'interior d'objectes, com per exemple parts del cos humà.

- Realitzar experiments amb dos o més elements virtuals que interaccionen en RA.

- Accessibilitat a recursos i espais públics.
- Simulacions de sistemes a través de RA.
- Traduccions en temps real.
- Accessibilitat a recursos i espais públics.

Les possibilitats son immenses i els docents poden recurrir a la RA en qualsevol nivell educatiu i assignatura, tant les relacionades amb les ciències (com per exemple matemàtiques o biologia) com les lletres (Història, Art, Llengua i Literatura), sempre i quan els continguts s'adaptin per a que l'aprenentatge resulti efectiu. A la botiga d'aplicacions d'IOS i Android en podem trobar nombroses aplicacions de RA i que han tingut una gran repercussió.

Un exemple és Star Walk, una aplicació enfocada a l'astronomia que que permet explorar més de 200.000 cossos celestes i obtenir-ne informació d'ells. Per activar-los s'ha d'inicialitzar l'aplicació i apuntar cap al cel on es veuran les estrelles, planetes, satèl·lits i constel·lacions en el seu lloc adequat des de la ubicació de l'usuari. A mesura que l'usuari mou el dispositiu, el mapa d'estrelles s'actualitza en temps real.

¿I qui no ha passat per aquella etapa de secundaria en la que ens havíem d'aprendre les cèl·lules? QuiverVision és una aplicació que subministra esquemes de les cèl·lules eucariota animal i vegetal. Una vegada impresos, els alumnes els pinten i posteriorment enfoquen amb l'aplicació al paper. Llavors una projecció en tres dimensions de la cèl·lula apareix en la pantalla, tot indicant les parts d'aquesta.

Layar es una aplicació de caire més generalista que permet escanejar publicacions i qualsevol material imprès com llibres, cartells o postals i assignar elements de Realitat Virtual sobre aquests. Un dels efectes que es pot aconseguir amb aquesta app és que les imatges d'un llibre cobrin vida reproduint-ne vídeos a sobre. El ministeri de cultura va desenvolupar continguts per aquesta aplicació que ens permet localitzar a qualsevol ciutat de l'estat més de 8000 biblioteques, i ens proporciona informació dels seus serveis, recursos i accés al catàleg quan apuntem fins a on està l'edifici físic.

Tal i com podem apreciar l'impacte que aquestes millores pot generar sobre diferents processos formatius i educatius és molt gran. Incloure tècniques innovadores com aquestes ens permet mesurar dit impacte i començar a adaptar-nos a un futur tecnològic que cada cop està més aprop. En el pròxim capítol estarem parlant sobre el World Mobile City Project, una iniciativa que combina la RA amb la geolocalització per ensenyar als més joves a moure's per la ciutat mentre s'entretenen.

1.2. Geolocalització.

Entenem per geolocalització la capacitat d'obtenir en temps real la ubicació geogràfica d'un dispositiu. Aquesta eina, que es va incorporar per primera vegada a un telèfon mòbil al 1999 amb la sortida del Benefon esc!, donaria lloc més tard al 2005 a la creació de l'imprescindible i gegant Google Maps. I es que els serveis de geolocalització ja formen part del nostre dia a dia. Són la nostra guia quan volem arribar a un lloc i no hi sabem anar-hi. També ens permeten accedir a informacions meteorològiques amb un sol clic, actualitzar la hora dels nostres smartphones i fins i tot consultar a quant ens cau el restaurant més proper. ¿Però com funciona tot aquest sistema?

Per contestar aquesta pregunta primer hem d'entendre el concepte de GPS. Les sigles fan referencia a Global Positioning System i es tracta d'un sistema de navegació per ràdio que utilitza les senyals proporcionades per una xarxa de 28 satèl·lits amb la finalitat de proporcionar informació sobre la ubicació i el temps a qualsevol programari que necessiti utilitzar-lo.

El mètode per obtenir la informació s'anomena trilateració i funciona de la següent manera: Quan el nostre telèfon rep una senyal d'un satèl·lit pot saber a quina distància es troba d'aquest ja que en coneix el temps que ha trigat la senyal en arribar i la seva velocitat. Ara bé, no sabem en quin punt exactament està ja que són infinits els punts que es troben a una mateixa distància. Amb la presència de tres senyals aquest problema es resol tal i com es pot veure a la **Fig 1.2.** Un quart satèl·lit es necessita per saber-ne l'altura també.

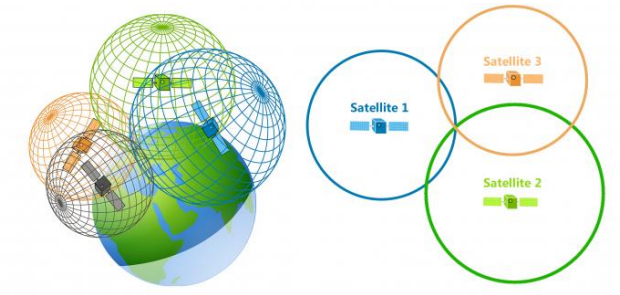

Fig. 1.2. Esquema de funcionament GPS.

Però aquest sistema suposa un consum d'energia molt gran pels nostres dispositius, i a no ser que s'utilitzi constantment, pot prendre fins a un minut cada vegada que s'obtinguin noves dades. Per aquest motiu aquest sistema es recolza sobre una altra tecnologia anomenada AGPS (Assisted GPS), que utilitza torres de telefonia per reduir el temps d'obtenció i millorar la localització.

Quan s'encén el receptor GPS, el mòbil envia a un servidor extern la identificació de l'antena on està connectat; i el mòbil obté com a resposta els satèl·lits situats a sobre de la seva posició. Aquestes dades es troben emmagatzemades al servidor extern, encara que també és habitual que el mòbil (o un altre dispositiu) descarregui un fitxer actualitzat amb les dades de posició A-GPS.

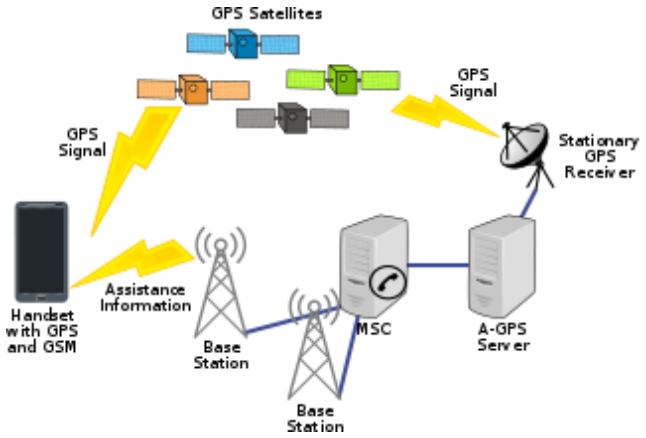

Fig. 1.3: Sistema Assisted-GPS.

1.3. Programari.

Per al desenvolupament de l'aplicació utilitzarem els següents entorns de desenvolupament : Unity 3d, Vuforia i Xampp.

1.3.1. Unity 3D.

Unity3D és un potent motor 3D multiplataforma que permet crear jocs i aplicacions en tres dimensions per a dispositius mòbils, equips d'escriptori, aplicacions web i videoconsoles. Aquesta plataforma consta d'un editor format per diferents finestres que es poden organitzar de forma personalitzada i que són la base per crear qualsevol aplicació (finestra del projecte, finestra d'escena, finestra de jerarquia, finestra de l'inspector, finestra d'eines i finestra de joc). A la **Fig 1.4** es pot apreciar la interfície de Unity.

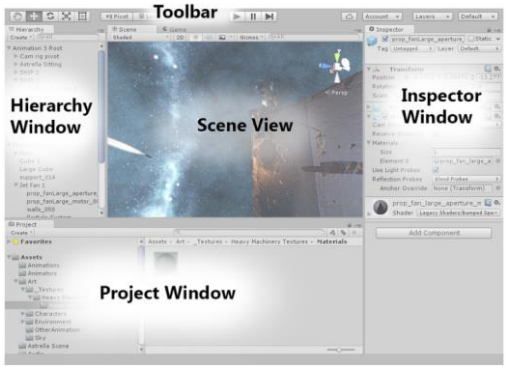

Fig. 1.4. Interficie de Unity.

La finestra del projecte mostra la biblioteca d'actius disponibles per al projecte que s'estigui desenvolupant. Quan importem actius al nostre projecte, apareixen aquí.

La finestra d'escena permet navegar visualment i editar l'escena en la que estem treballant . La vista d'escena pot mostrar una perspectiva 3D o 2D, segons el tipus de projecte en què estem treballant.

La finestra de la jerarquia és una representació de text jeràrquica de cada objecte de l'escena. Cada element de l'escena té una entrada a la jerarquia, de manera que les dues finestres estan inherentment enllaçades. La jerarquia mostra l'estructura de l'adhesió dels objectes entre si.

La finestra de l'inspector permet veure i editar totes les propietats de l'objecte seleccionat actualment. Com que diferents tipus d'objectes tenen diferents conjunts de propietats, el disseny i el contingut de la finestra de l'inspector varien.

La barra d'eines proporciona accés a les funcions més essencials. A l'esquerra conté les eines bàsiques per manipular la vista d'escena i els objectes que hi ha dins. Al centre hi ha els controls de reproducció, pausa i pas. Els botons de la dreta ens donen accés als nostres serveis Unity Cloud i al compte, seguits d'un menú de visibilitat de capa i, finalment, un menú de disseny d'editor (que proporciona alguns dissenys alternatius per a les finestres de l'editor i ens permet desar la nostra pròpia configuració).

A Unity tot succeeix en escenes. La pantalla del títol de la nostra aplicació pot ser una escena, els crèdits finals una altra, el joc en si pot ser una altra escena o varies, el menú principal una altra.. i totes les escenes juntes conformen l'aplicació final. Cada escena està formada per objectes: Càmeres, fonts de llum, reproductors, mapes en escena, personatges, entorns per on es mouen els personatges i multitud d'objectes més que fan possible jugar a un joc tal i com ho fem. Aquests objectes estan formats per components que els hi diuen el que han de fer i quan a cadascun d'ells. La creació d'aquests components requereix escriure codis, anomenats scripts, i en el nostre cas ho farem en llenguatge C# utilitzant l'aplicació de Visual Studio. També hi ha components predissenyats disponibles a la finestra de l'inspector. Els components han d'estar enllaçats als objectes sobre els quals volem aconseguir un determinat efecte.

És com una obra de teatre interactiva gegant, plena d'escenes i objectes que exerceixen diferents funcions i que tots junts, fan possible l'experiència viscuda pels espectadors.

1.3.2. Vuforia.

Vuforia és una plataforma de desenvolupament d'aplicacions de RA i Realitat Mixta (RM) multiplataforma, amb un sistema de tracking robust que funciona amb una gran varietat de hardware (incloent dispositius mòbils com smartphones i tauletes o dispositius Head-Mounted Display com les famoses Microsoft HoloLens). La integració de Unity en Vuforia permet crear aplicacions i jocs de visió per Android i iOS de forma dinàmica.

Aquesta plataforma permet crear aplicacions tant basades en marcadors com sense marcadors. Per tal de poder utilitzar-la i beneficiar-nos de les possibilitats de Vuforia, el primer que hem de fer és importar el paquet de desenvolupament software (SDK) de Vuforia a Unity. En les últimes versions de Unity és possible importar aquest paquet en el mateix moment que activem els diferents paquets de la instal·lació.

Vuforia proporciona interfícies de programació d'aplicacions (API) en C ++, Java, Objective-C++ (un llenguatge que utilitza una combinació de sintaxi C ++ i Objective-C) i els llenguatges .NET mitjançant una extensió del motor de joc Unity. D'aquesta manera, el SDK suporta tant el desenvolupament natiu per iOS com Android, mentre que també permet el desenvolupament d'aplicacions RA a Unity que són fàcilment portàtils a ambdues plataformes. Les aplicacions RA desenvolupades amb Vuforia són, per tant, compatibles amb una àmplia gamma de dispositius mòbils, incloent els telèfons i taulers iPhone, iPad i Android que funcionen amb Android OS 2.2 o superior i un processador ARMv6 o 7 amb capacitats de processament FPU (Floating Point Unit).

En el nostre projecte, farem una aplicació basada en marcadors utilitzant els corresponents elements de Vuforia. El primer objecte que es necessita tenir activat a Unity és una càmera de RA, que s'encarrega d'identificar l'activador dels elements virtuals i de fer el corresponent tracking una vegada identificat. Per activar vuforia ens hem de crear un compte a la web de Vuforia Developer i copiar la clau de llicència en la configuració de Vuforia de Unity.

Un dels components més importants de Vuforia, i que al nostre projecte utilitzarem son els Image Targets. Els image targets representen imatges que Vuforia Engine pot detectar i rastrejar. A diferència dels marcadors tradicionals, els codis de matriu de dades i els codis QR, els Image Targets no necessiten regions o codis blancs i negres especials per ser reconeguts. El motor de Vuforia detecta i rastreja les característiques que es troben naturalment a la imatge mateixa comparant aquestes característiques naturals amb una base de dades que prèviament hem importat a Unity. Un cop detectat l'objectiu d'imatge, Vuforia Engine rastrejarà la imatge sempre que estigui almenys parcialment al camp de la càmera. Des del portal de Vuforia podem crear aquestes bases de dades d'Image Targets amb el nostre compte.

Es recomana que siguin rics en detalls, que tinguin un bon contrast i que no segueixen patrons molt repetitius. A la **Fig 1.5** podem veure un exemple del que seria un target fàcilment reconegut pel sistema.

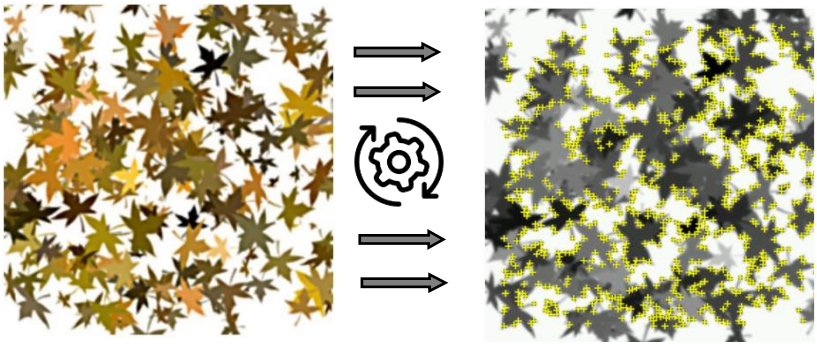

Fig. 1.5. Exemple d'un Image target ben dissenyat.

Però al capítol d'implementació veurem més en detall com hem introduït les funcionalitats de Vuforia a la nostra aplicació.

1.3.3. XAMPP.

Xampp és un servidor independent de plataforma, que incorpora la base de dades MySQL, el servidor Web Apache i els intèrprets per llenguatges de Script: PHP i Perl. El nom prové de l'acrònim X (per qualsevol dels diferents sistemes operatius), Appache, MySQL, PHP i Perl. El programa és lliure, sota la llicència de GNU General Public License i actua com a servidor Web lliure i fàcil d'utilitzar.

Oficialment, els dissenyadors de XAMPP només pretenien el seu ús com una eina de desenvolupament, per permetre als dissenyadors de llocs webs i programadors testejar el seu treball en els seus propis ordinadors sense cap accés a Internet. A la pràctica, però, XAMPP és utilitzat actualment com a servidor de llocs web i, amb algunes modificacions, és generalment prou segur per ser-ho.

Nosaltres utilitzarem XAMPP per implementar un servidor amb el que ens connectarem des de l'aplicació. Concretament l'utilitzarem per enviar i descarregar els fitxers que contenen la informació de les proves de la gimcana. També crearem una base de dades on s'enviaran i s'emmagatzemaran els resultats obtinguts de cada grup el dia de l'activitat.

Capítol 2. World Mobile City Project.

El World Mobile City Project (WMCP) és un projecte impulsat per l'equip LaceNet[4] que combina diferents tecnologies com internet, la geolocalització i la RA, amb la col·laboració de diferents centres entre els quals estem nosaltres, l'Escola d'Engenyeria de Telecomunicacions i Aeroespacial de Castelldefels. El MWCP proposa una activitat de georeferenciació amb l'objectiu d'aprendre a localitzar qualsevol punt de la ciutat combinant mitjans clàssics com mapes i targes, i mitjans més moderns com les tecnologies mencionades anteriorment, aplicant el treball cooperatiu en petits grups.

Aquesta activitat està dirigida a l'alumnat dels diferents centres de secundària, amb el suport del professorat. El projecte s'implementa a diverses ciutats i pobles dels Països Catalans i de l'Estat espanyol: Alcoi, Barcelona, Camp de Morvedre, Castelló, Elx, Gasteiz, Igualada, Manresa, València, Vic, Vall dels Alcalans, Valls i Xàtiva. El nom de l'activitat normalment rep el nom de la ciutat més el sufix *ada*. Per exemple, la versió que es duu a terme a Barcelona és la Barcelonada.

L'activitat es presenta en forma de joc. En el cas de la Barcelonada, els alumnes que hi participen es desplacen fins la plaça de Catalunya, on s'inicia la gimcana, acompanyats dels seus professors. La gimcana es realitza en grups de 4 alumnes, així que s'han d'organitzar abans de començar. També s'han d'haver descarregat prèviament una aplicació que serà el principal motor d'aquesta activitat. Amb aquesta aplicació, els estudiants tindran accés a les diferents proves que conformen la gimcana, però abans que tot s'han de registrar com a grup i escola. Una vegada fets els anteriors passos, cada grup rep un número determinat de targets corresponents a diferents llocs d'interès de la ciutat que contenen la informació necessària per visitar l'indret, i comença l'activitat.

Els alumnes s'han de desplaçar fins als diferents punts de la ciutat indicats en els targets repartits. Depenent de la modalitat que hagin escollit els creadors de la gimcana, hauran de visitar els llocs en un ordre específic, o ho podran fer arbitràriament. Una vegada arribin a les corresponents localitzacions, hauran d'obrir l'aplicació i enfocar amb la càmera que incorpora als targets.

Quan l'aplicació els detecti apareixeran una sèrie de cubs a sobre en forma de RA. Aquests cubs corresponen a les diferents proves disponibles en aquella localització. Una vegada més, depenent de la modalitat del joc, els alumnes hauran de realitzar només una acció diferent a cada target, o totes, ja que l'aplicació (la que farem) està pensada per poder utilitzar-se de forma flexible.

Una vegada el grup hagi realitzat i completat les accions corresponents al primer target, es dirigiran al lloc indicat pel següent target, i així successivament fins que acabin totes les proves. Els participants aniran acumulant punts a mesura que

[4] Associació de professionals de l'ensenyament de la comarca del Bages, interessats per l'ús educatiu de la telemàtica. l'Equip LaceNet ofereix projectes telemàtics als centres educatius. Al llarg d'aquests anys s'ha anat diversificant l'oferta i ampliant el marge d'edat dels nois i noies als quals van destinats.

vagin avançant, i podran anar comprovant les proves que ja han fet i les que encara no a través d'un panell de seguiment que tenen disponible a l'app.

A l'aplicació que nosaltres dissenyarem, hi ha quatre tipus de proves disponibles, que es veuran a l'apartat d'especificacions de l'aplicació. Cal esmentar que el codi que executa el joc està pensat per poder introduir nous tipus d'accions en un futur sense haver de canviar gaires coses.

Resumidament, el WMCP és un projecte d'inclusió digital i l'excusa perfecta per conèixer la ciutat multidisciplinàriament, aprendre a aprendre i assolir un gran nombre de competències cooperativament, de forma lúdica. Al document *Guia didàctica de la Barcelonada* de la web del MWCP es pot trobar més informació.

2.1.1. Objectius del WMCP.

L'objectiu principal del World Mobile City Project és aprendre a desplaçar-se amb autonomia per la ciutat, fent-ne us de les últimes tecnologies que incorporen els smartphones. Així mateix també es marca com a objectius:

- Facilitar l'autonomia de l'alumnat. Millorar la competència digital dels estudiants.
- Compartir coneixement a la Xarxa.
- Conèixer els llocs d'interès de la ciutat (patrimonials, culturals, socials..).
- Localitzar qualsevol punt de la ciutat amb l'ajuda de dispositius mòbils.
- Dissenyar rutes geolocaitzades amb els punts triats de la ciutat.
- Familiaritzar-se amb l'ús de les darreres tecnologies .
- Moure's per la ciutat fent servir el transport adient.
- Formar criteri en la utilització dels dispositius mòbils i de les aplicacions.
- Fomentar el civisme i bons hàbits ciutadans.
- Construir nou coneixement personal mitjançant estratègies de tractament de la informació amb suports digitals
- Donar a conèixer les possibilitats de la RA aplicada a l'ensenyament

2.1.2. Competències bàsiques.

Com ja hem dit anteriorment, un dels objectius d'aquest projecte és aconseguir que els joves assoleixin un gran nombre de competències mitjançant aquesta activitat. La principal competència bàsica de caire metodològic és la d'aprendre a aprendre. Els permetrà seguir aprenent de forma autònoma d'acord amb els propis objectius i necessitats així com guiarà les accions i el desenvolupament de totes les altres competències bàsiques.

En la **Taula 1** es poden apreciar quines són aquestes competències.

Taula 1: Competències del projecte.

Competències bàsiques de l'àmbit social	
Dimensió cultural i artística	
Competència 8	Analitzar les manifestacions culturals i relacionar-les amb els seus creadors i la seva època, per interpretar les diverses cosmovisions i la seva finalitat.
Competència 9	Valorar el patrimoni cultural com a herència rebuda del passat, per defensar-ne la conservació i afavorir que les generacions futures se l'apropiïn
Competència 10	Valorar les expressions culturals pròpies, per afavorir la construcció de la identitat personal dins d'un món global i divers.
Competències bàsiques de l'àmbit digital	
Dimensió instruments i aplicacions	
Competència 3	Utilitzar les aplicacions bàsiques d'edició d'imatge fixa, so i imatge en moviment per a produccions de documents digitals.
Dimensió de tractament de la informació i organització dels entorns de treball i aprenentatge	
Competència 4	Cercar, contrastar i seleccionar informació digital adequada per al treball a realitzar, tot considerant diverses fonts i mitjans digitals.
Dimensió comunicació interpersonal i col·laboració	
Competència 7	Participar en entorns de comunicació interpersonal i publicacions virtuals per compartir informació.
Competència 8	Realitzar activitats en grup tot utilitzant eines i entorns virtuals de treball col·laboratiu.
Competències bàsiques de l'àmbit lingüístic	
Dimensió comprensió lectora	
Competència 2	Aplicar estratègies de comprensió per obtenir informació, interpretar i valorar el contingut d'acord amb la tipologia i la complexitat del text i el propòsit de la lectura.
Competència 3	Utilitzar, per comprendre un text, l'estructura i el format de cada gènere textual i el component semàntic de les paraules i de les estructures morfosintàctiques més habituals
Competències bàsiques de l'àmbit de cultura i valors	
Dimensió sociocultural	
Competència 7	Comprendre i valorar el nostre món a partir de les arrels culturals que l'han configurat

Capítol 3. Especificacions de l'aplicació.

Tal i com hem vist en el capítol anterior, un dels factors clau per l'execució del World Mobile City Project és l'existència d'una aplicació que gestioni tota l'activitat el dia que els estudiants surten al carrer per fer la gimcana. Amb aquest aplicatiu els alumnes reben instruccions de les proves que han de realitzar, que algunes es fan dins de la mateixa, i d'altres fent-ne ús d'altres aplicacions. Actualment existeix una aplicació disponible però es requereix una nova versió que agilitzi la creació d'activitats. Per tant amb la nova aplicació es pretén cobrir la necessitat d'un sistema més dinàmic. Cal esmentar que aquesta aplicació que es demana no és una versió final si no que ha d'estar pensada per afegir noves funcionalitats de forma senzilla i sense haver de canviar l'arquitectura per afegir canvis.

3.1. Audiència.

Els usuaris potencials d'aquesta aplicació seran estudiants, professors i la resta de membres directament relacionats amb l'educació secundaria. Cal especificar que estudiants i professors no utilitzaran l'aplicació de la mateixa forma, doncs tindran diferents rols, i per tant diferents perfils.

3.2. Situació actual.

El model d'aplicació que s'ha utilitzat fins ara el dia de la gimcana ja porta carregades les proves que els alumnes han de realitzar, de manera que els alumnes se la descarreguen, s'identifiquen com a escola i ja poden iniciar la partida. En aquesta aplicació no es distingeixen diferents tipus d'usuaris, doncs només hi ha un, l'usuari final, és a dir l'estudiant. Amb aquest model cada vegada que es vol organitzar una activitat s'ha de crear una nova aplicació, i per tant, l'administrador ha de tenir coneixements informàtics i saber moure's per la interfície de Unity, que es el programa utilitzat pel desenvolupament d'aquesta.

3.3. Objectius de l'aplicació.

L'objectiu és aconseguir una aplicació amb la que es pugui tant crear i gestionar activitats, com jugar-les. Resumidament, una aplicació que contingui dos perfils d'usuari, un d'administrador, i un altre de jugador, on l'administrador no necessiti dominar llenguatges informàtics i es pugui focalitzar directament en l'activitat com a tal. Una aplicació reutilitzable, i que a la vegada permeti realitzar canvis i afegir noves funcionalitats en aquesta sense haver de canviar-ne el model base.

3.4. Domini del problema.

El domini del problema contempla la RA, la geolocalització, internet, l'educació, l'entreteniment, l'autoaprenentatge, servidors web i bases de dades.

3.5. Requeriments de l'aplicació.

Es requereix una aplicació amb dos tipus d'usuaris, un de professor, que serà l'administrador de l'activitat, i un altre d'estudiant, usuari que jugarà la partida creada pèviament. Per aquesta proposta, també es requereix la presència d'un servidor en el qual s'emmagatzemaran els arxius de cada partida creada, i al qual s'accedirà quan els estudiants hagin de carregar la partida. A la **Fig 3.1** podem observar un esquema general de la relació entre aquests tres elements.

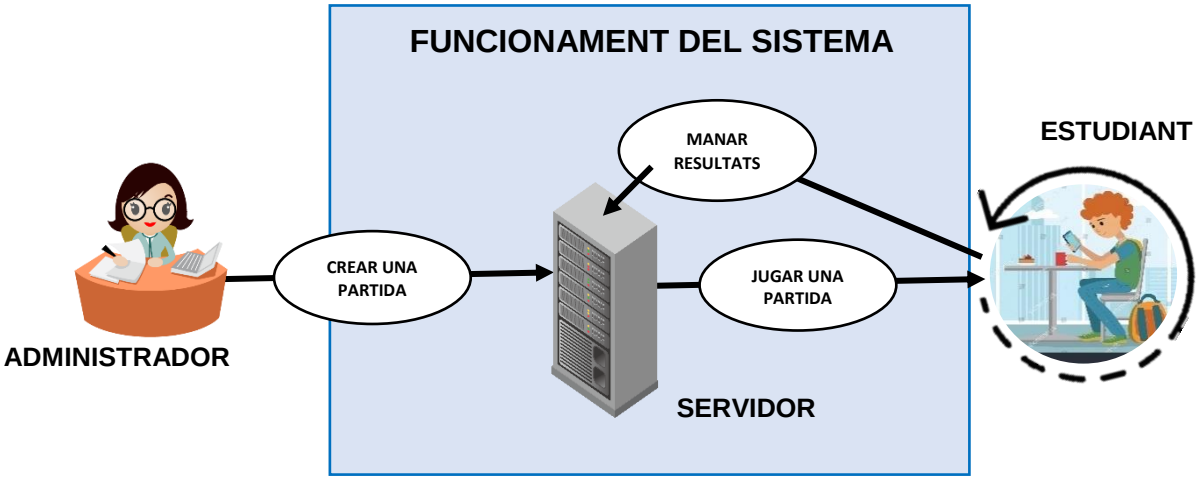

Fig. 3.1 Visió general del sistema.

A l'obrir l'aplicació s'haurà de carregar un menú principal que incorpori dues opcions, accedir al perfil d'estudiants o accedir al perfil de l'administrador, tal i com es pot veure a la **Fig.3.2.**

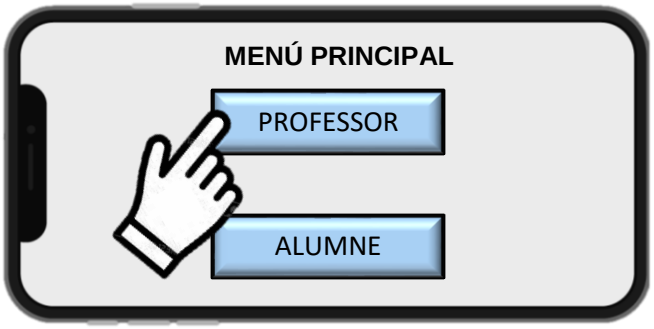

Fig. 3.2. Menú principal.

El menú del professor servirà per generar una nova activitat. Cada activitat estarà definida per quatre variables principals:

- **Número de targets:** L'administrador decidirà quantes localitzacions contindran proves durant l'activitat. Cada localització estarà representada amb un target que l'alumne haurà d'enfocar amb la càmera de RA per

desbloquejar la prova. Hi haurà fins a 20 targetons disponibles, però en un futur es podria ampliar sense cap problema.

- **Número d'accions:** És el número de proves que hi haurà a cada localització. Cada target podrà contenir fins a 6 proves.

- **Selecció entre una prova o totes:** L'administrador haurà de decidir si vol que els alumnes facin només una vegada cada prova o si pel contrari les hauran de fer totes.

- **Selecció de l'ordre de l'activitat:** L'administrador haurà de decidir si vol que els alumnes recorrin les diferents localitzacions de forma ordenada o arbitràriament.

Hi haurà 4 tipus de proves disponibles, i el sistema de gestió de proves haurà d'estar pensat per poder afegir més proves en un futur sense canviar l'arquitectura de l'aplicació. Les proves seran les següents:

- **Acció de text:** L'administrador escriurà un missatge dirigit l'estudiant per demanar que faci alguna acció en concret, com per exemple fer una foto o vídeo del lloc i penjar-ho en alguna xarxa social, o qualsevol altre acció que es vulgui.

- **Acció de text + imatge:** L'administrador haurà de seleccionar una fotografia diferent per cada target i escriure una pregunta comú per tots ells, que l'estudiant haurà de respondre.

- **Formulari:** L'administrador haurà d'escriure el link al corresponent formulari a contestar.

- **Trivial:** L'acció de trivial estarà formada per preguntes i cada pregunta tindrà quatre respostes, de les quals només una serà la correcta. Cada pregunta anirà associada a un target. És a dir, la primera pregunta correspondrà al primer target, la segona al segon i així successivament.

Totes les partides generades estaran identificades amb un codi de quatre dígits numèrics que el creador de l'activitat haurà de decidir. Aquest codi servirà per registrar una partida i que l'alumne pugui accedir a ella simplement introduint-lo.

Haurà de ser possible preparar les proves de dues maneres diferents:

- **De forma manual:** l'administrador anirà creant les proves que conformin l'activitat una per una, escollint entre opcions, desplegables, botons i entrades de text. Cada vegada que s'hagi acabat de preparar una prova es manarà un fitxer de text corresponent a la prova al servidor. A la **Fig 3.3** es pot veure una idea d'aquest mode.

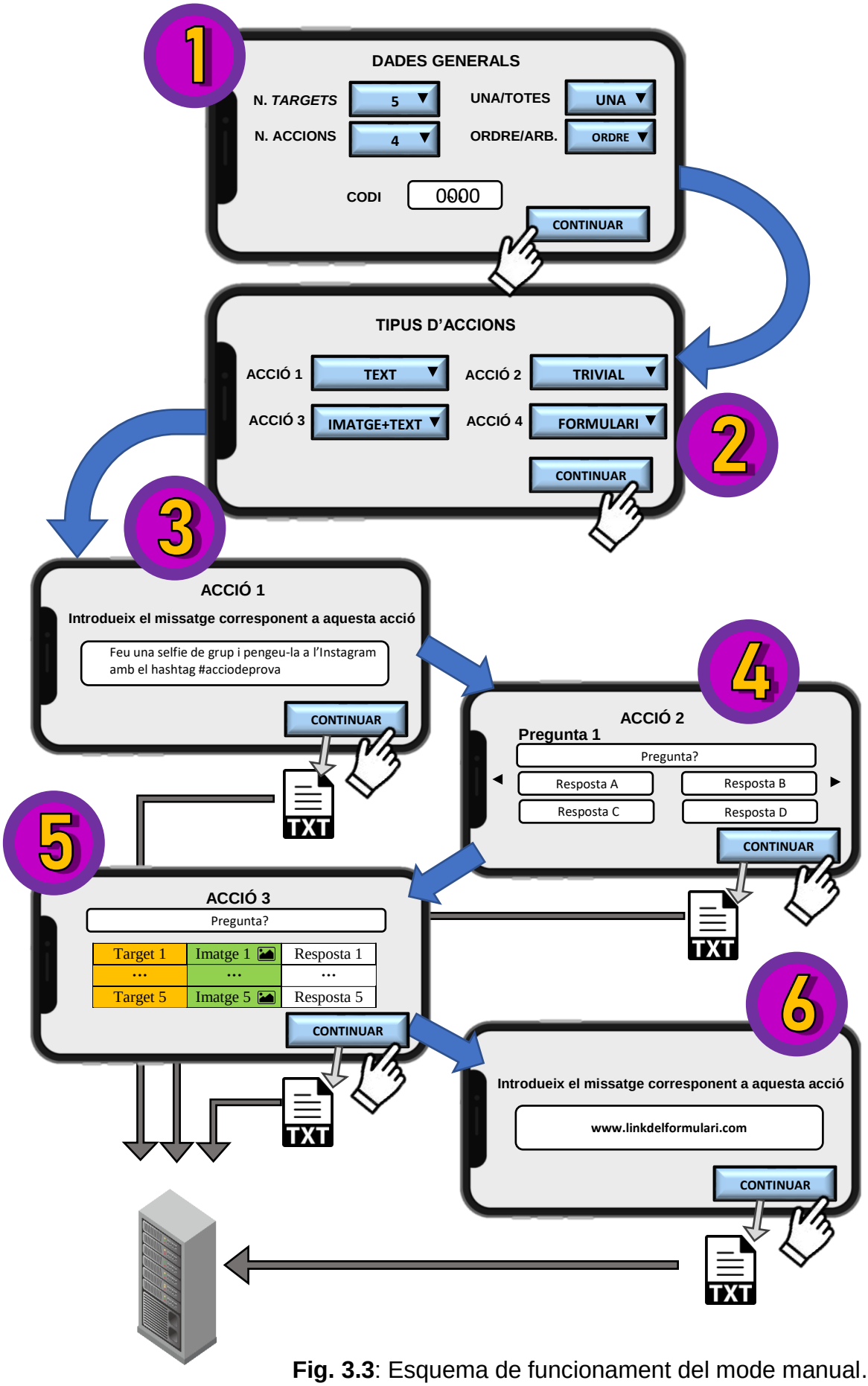

Fig. 3.3: Esquema de funcionament del mode manual.

- **Carregant fitxers directament:** En aquest cas, l'administrador ja coneix com ha de ser l'estructura dels fitxers de text i l'única cosa que ha de fer és anar pujant-los des del corresponent menú. La idea és crear un panell on es puguin anar carregant els fitxers per després manar-los al servidor. A la **Fig 3.4** es pot veure un esquema d'aquest model.

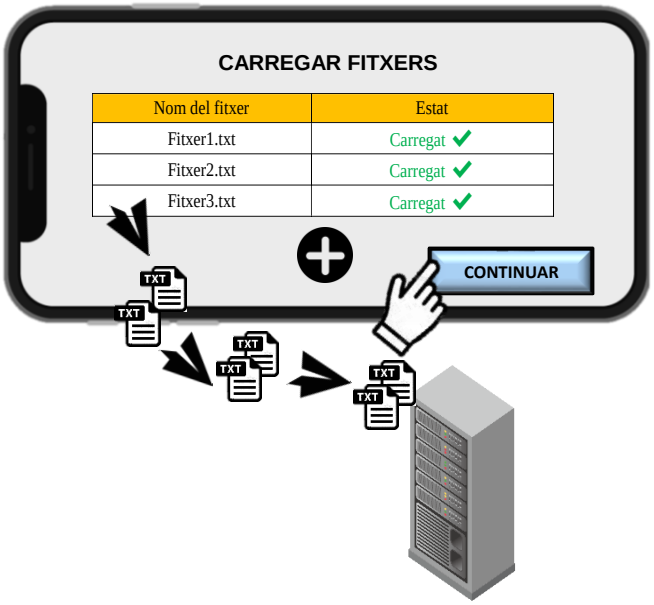

Fig. 3.4 Model de càrrega de fitxers.

El perfil de l'alumne servirà per poder realitzar l'activitat preparada per l'administrador. En accedir a aquest perfil l'usuari s'haurà de trobar un menú en el qual pugui identificar-se com a escola i com a grup. Per carregar la partida hauran d'escriure el codi de 4 dígits numèrics que prèviament el professor els haurà facilitat.

Fig. 3.5 Model per carregar la partida.

Una vegada introduït el codi, es descarregaran tots els arxius necessaris del servidor i es carregarà la corresponent activitat. L'usuari llavors accedirà a una altre pantalla on veurà el que la càmera del seu dispositiu (que s'activarà) està

veient. Quan enfoqui amb la càmera algun dels targets de l'activitat, les proves a realitzar es manifestaran com a cubs de RA sobre els targets, tal i com es pot veure a la **Fig 3.6.**

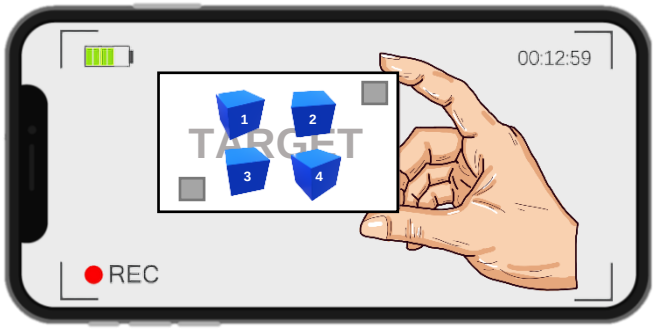

Fig. 3.6: Vista d'un target enfocat.

La idea és que quan l'usuari polsi sobre algun dels cubs s'obri un panell que contingui la prova associada a aquell cub. Per aquesta proposta es necessitaran una sèrie de panells base per cada tipus de prova, que s'ompliran amb l'informació continguda per cada cub. En el cas del formulari no hi haurà panell ja que el link s'ha d'obrir amb el navegador. En la **Fig 3.7** es pot observar un concepte del que serien els panells.

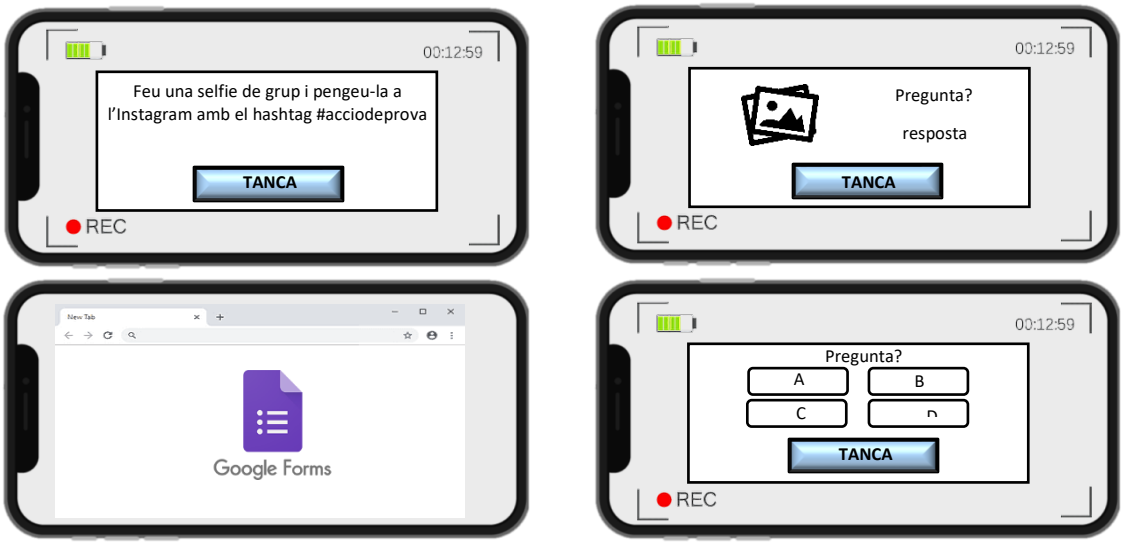

Fig. 3.7: Panells d'accions.

Capítol 4. Implementació de l'aplicació.

En aquest capítol veurem com hem anat desenvolupant l'aplicació en base a les especificacions requerides i quines tècniques hem utilitzat per solucionar els problemes que ens hem anat trobant al llarg del camí. Tal i com hem comentat en els anteriors capítols, per desenvolupar el nostre projecte hem utilitzat la plataforma d'Unity amb el SDK de Vuforia per la part de RA, i el llenguatge C# per programar els scripts[5].

A l'iniciar el projecte els meus coneixements sobre la plataforma de Unity eren nuls, per la qual cosa vaig començar amb un model d'aplicació que no incorporava totes les funcionalitats, però que em permetria agafar pràctica i aprendre a utilitzar les eines que Unity incorpora. Cal esmentar també, que aquesta primera versió tampoc complia amb els requisits demanats en les especificacions. A la **Fig 4.1** es mostren captures dels panells principals.

Fig. 4.1. Primera versió.

Hi havia una única escena en la que succeïa tot: Es generava una partida i seguidament es tancaven tots els panells per jugar. Les dades no s'emmagatzemaven en fitxers, ni s'utilitzava un servidor. Només hi havia un tipus de prova, la de text. Quan vaig agafar més pràctica em vaig adonar que aquella primera versió no tenia una base flexible i que estava malament dissenyada des d'un principi. Va ser en aquell llavors quan vaig decidir fer un projecte de Unity nou i començar de zero amb la segona versió.

La segona versió permetia generar partides a partir de fitxers que es carregaven i descarregaven d'una carpeta de la meva computadora, de manera que només es podia jugar des de la meva màquina. A la **Fig 4.2** es mostren captures dels panells principals. Aquesta versió no estava ben pensada per incorporar dos perfils d'usuari, així que vaig decidir començar de zero un altre cop amb la tercera

[5] L'script fa referencia a un document que conté instruccions, escrites en codi de programació, i que donaran vida a la nostra aplicació.

i última versió, que és en la que em centraré i explicaré amb més detall en aquest capítol.

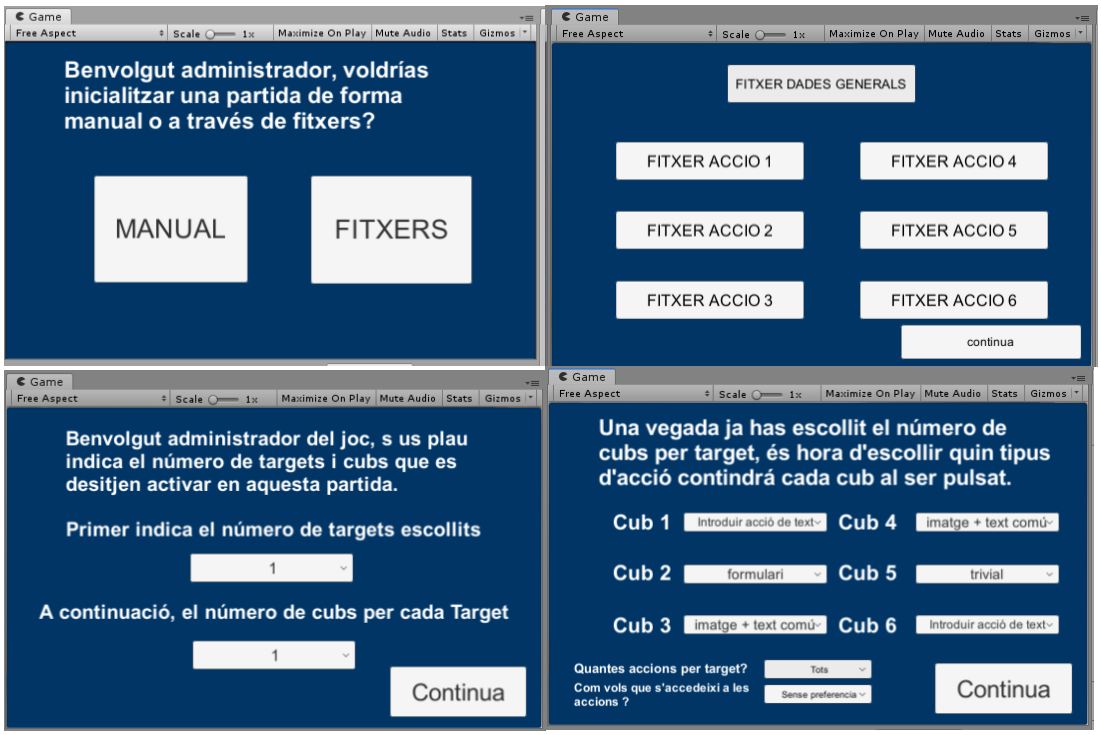

Fig. 4.2. Primera versió.

El primer que he fet en la tercera versió ha sigut crear tres escenes diferents, una corresponent al menú principal de l'aplicació, una altra pel perfil de l'administrador, i una altra pel perfil de l'usuari. Seguidament he creat dos scripts, un anomenat *gamemanag()*, i l'altre t*eachermanager()*, on aniré escrivint la majoria de funcions que gestionen els diferents panells, objectes i components de les escenes. A la **Fig 4.3**. es pot observar el disseny del menú principal, amb dos botons per accedir als dos tipus d'usuari possibles.

Fig. 4.3. Menú principal.

Per tal de poder accedir a les dues escenes, he creat un script amb una funció anomenada *loadscene()* que té com a entrada un enter. Depenent de si és un 0 o un 1, carrega una escena o l'altre. Després he adjuntat dos botons de manera que si premem a *Soc un@ professor@*, o *soc un@ alumn@* es carreguen les corresponents escenes. En la següent secció veurem com hem implementat el servidor que emmagatzemarà es arxius necessaris pel correcte funcionament de l'aplicació.

4.1. Servidor.

Tal i com ja vam veure a les especificacions de l'aplicació, la idea era que tota la informació de les activitats creades pels administradors quedés registrada en fitxers de text dins un servidor públic. D'aquesta manera, els alumnes poden descarregar-se les activitats introduint només el codi que el professor els hi facilita en el dia de la gimcana. Aquest servidor també servirà per crear la connexió amb una base de dades MySql (anomenada *resultats*) on es registraran les puntuacions dels diferents grups participants de les gimcanes.

Per implementar aquest servidor la primera cosa que he fet ha sigut agafar un ordinador disponible de casa, accedir a la configuració del meu enrutador d'Orange i obrir el port 8523 (la idea era utilitzar el port 80 però l'aplicació d'Skype ja estava utilitzant-lo i donava problemes).Seguidament he instal·lat el programa Xampp i he obert el configurador per canviar la configuració del servidor Apache, que escoltava pel port 80 i havia d'escoltar pel 8523.

Una vegada configurats els ports, he creat una carpeta dins la carpeta htdocs d'Xampp (carpeta on s'han d'emmagatzemar els elements del servidor) anomenada *miservidor*. A dins d'aquesta he creat dues carpetes més, una anomenada *imagenes*, i l'altre *ficheros.*

També he afegit quatre codis php per:

- introduir imatges desde Unity a la carpeta imagenes.
- introduir fitxers desde Unity a la carpeta ficheros.
- Crear la connexió amb la base de dades.
- introduir elements a la base de dades.

Els codis per gestionar els arxius es diuen *cargararchivo.php* (per carregar les imatges) i *cargararchivotexto.php* (per carregar els fitxers de text) tenen un funcionament molt senzill: Creen una copia dels arxius que reben a l'adreça del servidor adequada *(htdocs\ miservidor\ imagenes\ imatgedeprova.jpg) i (htdocs\ miservidor\ ficheros\ fitxerdeprova.txt).*

Els codis per gestionar la base de dades realitzen una connexió senzilla amb la base de dades *resultats*, per introduir a la taula *usuaris* les variables *número del grup, escola i puntuació* des de l'aplicació d'Unity.

A la **Fig 4.4** es pot veure una captura del servidor.

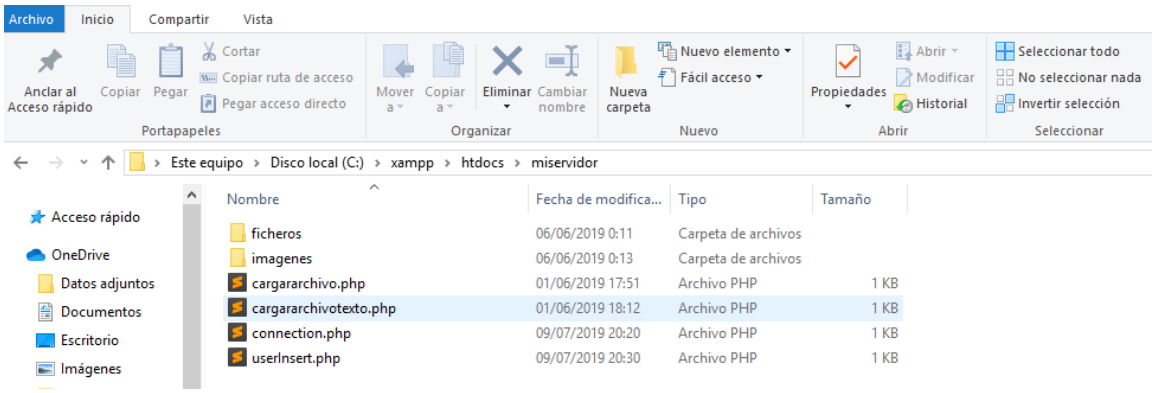

Fig. 4.4: Elements del servidor.

Finalment hem activat el servidor Appache i MySQL des del configurador (**Fig 4.5**) i el servidor ja està llest per utilitzar-se.

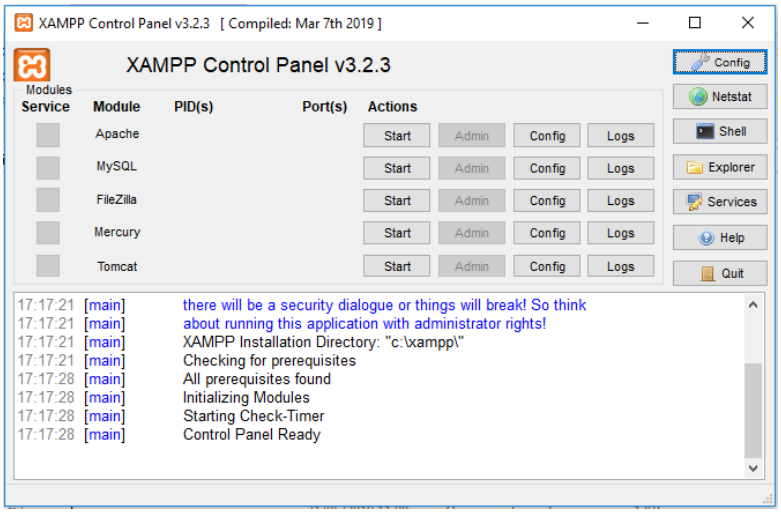

Fig. 4.5: Activació servidor.

4.2. Escena de l'administrador.

El perfil dels professors serveix per generar noves activitats, així que el primer element que he introduït a l'escena és un panell (**Fig 4.6**) que permet escollir entre dues opcions a través de dos botons:

- Crear una activitat de forma manual.
- Penjar fitxers al servidor directament.

Seguidament he creat els corresponents panells per crear una partida manualment i per pujar fitxers, i una funció anomenada *obrirmodemanualofitxers(*

) a l'script *teachermanager* per accedir als corresponents panells des dels botons.

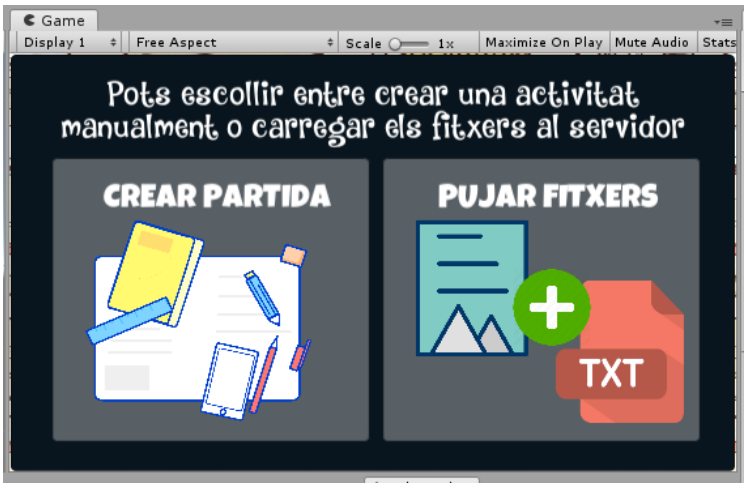

Fig. 4.6: Panell d'elecció.

4.2.1. Crear una partida manualment.

El primer panell que he creat es diu *dadesgen* i permet que l'usuari esculli els valors de les dades generals de la partida: El número de targets, el número d'accions per target, la possibilitat de fer una acció només una vegada o totes, l'ordre a seguir en l'activitat, el codi i les coordenades. L'elecció de cadascuna de les variables es fa mitjançant desplegables (dropdowns) excepte el codi, que funciona amb un camp d'entrada. Per tal d'emmagatzemar aquests valors he creat una classe Singleton[6] anomenada *clasegenerica* i he declarat quatre variables públiques de tipus enter.

Fig. 4.7: Elecció variables principals.

[6] Una classe Singleton és una classe que només permet ser instanciada una sola vegada, de manera que la primera vegada que la instanciem es crea una de nova, i les posteriors retorna la que ja hi ha.

Per tal d'introduir les coordenades de les diferents localitzacions he afegit un botó que obre el panell coordenades. Aquest panell compta amb una scroll view dins la qual es carreguen els adients camps d'entrada per introduir la latitud i la longitud. La funció encarregada de carregar els camps es diu *activarpanellcoordenades(),* i la funció *tancarpanellcoordenades()* s'encarrega d'enviar les latituds i longituds a la *matriucoordenades* de la *clasegenerica*, i seguidament tancar el panell. A la **Fig. 4.8** es pot veure el panell.

Fig. 4.8. Panell de coordenades.

Una vegada fet això, he afegit una funció al *teachermanager*. Aquesta funció es diu *introduirgenerals* i s'executa prenent el botó de *comencem* que he introduït al panell. Llegeix els valors de les variables i s'encarrega de fer dues coses:

- Guardar els valors a la *clasegenerica*.
- Generar un fitxer de text amb els valors i manar-lo al nostre servidor. Els fitxers de dades generals tenen la següent estructura:

  (1) Primera línia: Codi identificador de l'activitat. Ha de ser un codi numèric de 4 dígits

  (2) Segona línia: Número de targets (localitzacions) que contindran activitats per realitzar el dia de la prova.

  (3) Tercera línia: Número de proves que hi haurà a cadascuna de les localitzacions.

  (4) Quarta línia: Serà un 0 o un 1. Un 0 indica que només s'haurà de fer una acció de cada tipus una sola vegada. Un 1 significa que s'hauran de fer totes.

  (5) Cinquena línia: Serà un 0 o un 1. Un 0 indica que les localitzacions s'han de visitar de forma ordenada. Un 1 significa que es poden visitar arbitràriament.

(6) Coordenades de les localitzacions.

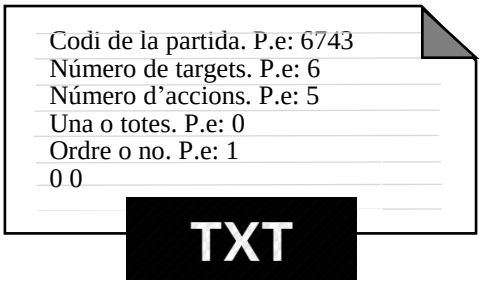

Fig. 4.9. Model de fitxer.

- Activar el següent panell.

El següent panell es diu *tipusdeaccions* i serveix per que l'administrador esculli quin tipus de prova contindrà cadascun dels cubs. El tipus d'acció (text, text+imatge, formulari, trivial) s'introdueix a través d'un desplegable i depenent el número de targets que s'hagin escollit prèviament apareixeran més o menys desplegables. També he creat una llista d'enters a la *clasegenerica* on es van afegint els valors escollits als desplegables (0,1,2 o 3) quan donem al botó de continua. A la **Fig 4.10** es pot veure aquest panell.

Fig. 4.10. Tipus d'accions.

El següent que he fet ha sigut dissenyar 4 panells que serveixen per organitzar les diferents proves de l'activitat. Els he agrupat tots en un objecte buit anomenat *constructors.* Comencem pel constructor d'accions de text.

4.2.1.1. Constructor de text.

Aquesta prova consisteix en realitzar el que l'administrador deixa encarregat en forma de missatge. Per aquesta proposta, he afegit un camp d'entrada al panell on l'administrador pot escriure l'acció a fer. Més a baix a l'esquerra he afegit un

botó en forma de caixa que serveix per carregar el material del corresponent cub. Aquest botó acciona una funció anomenada *carregarmaterial*(), que obre un quadre de diàleg per seleccionar la imatge que es vulgui com a material del cub i la mana al servidor. El botó de *continua* acciona dues funcions del *teachermanage*r:

- La funció *carregaracciotext()*, que s'encarrega de crear un arxiu de text corresponent a la prova i manar-lo al servidor. Aquests fitxers tenen la següent estructura:

  (1) Primera línia: paraula "TEXT". Aquesta paraula permetrà identificar el tipus d'acció quan l'alumne es connecti amb el servidor.

  (2) Segona línia: nom de la imatge del material: Serà necessari per carregar el material des del servidor.

  (3) Tercera línia: Missatge de text.

Fig. 4.11: Model fitxer text.

- La funció *seguentacció()*, que elimina el primer element de la llista de tipus d'accions de la *clasegenerica* quan ja s'ha creat la corresponent prova, i llegeix el primer element de nou per tal d'obrir el panell adient per la següent prova. Aquesta funció s'utilitza a tots els constructors.

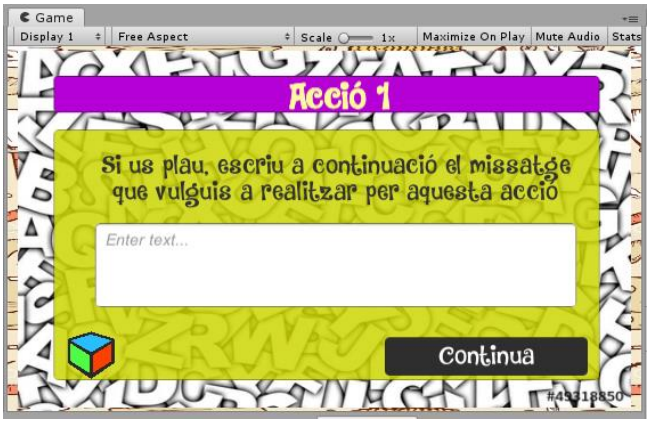

Fig. 4.12. Panell acció text.

4.2.1.2. Constructor imatge+text.

Pel disseny d'aquest constructor he optat per introduir un camp d'entrada i una scroll view. El camp d'entrada serveix per introduir la pregunta i la scroll view per afegir les imatges i respostes de cada target. He dissenyat la scroll view de tal forma que cada fila correspon a un target i està formada per quatre elements: el numeró de target, un botó per carregar la imatge, un quadre d'estat (pendent o carregat) i un camp d'entrada per introduir la corresponent resposta.

Aquests elements apareixen de forma automàtica quan es carrega el constructor i el codi responsable d'això es troba dins la funció *seguentaccio()* del *teachermanager*. El codi consulta quin és el número de targets a la *clasegenerica* i en funció d'això instancia els corresponents elements.

Tots els elements estan agrupats dins d'un objecte buit anomenat plantilles, per facilitar-ne el codi. He creat també dos vectors a la *clasegenerica*, un per guardar els noms de les imatges, i un altre per guardar les respostes. Els botons per carregar imatges utilitzen una funció anomenada *carregarboto().*

Aquesta funció obre un quadre de diàleg per seleccionar la imatge que es vulgui, i una vegada seleccionada, la mana al servidor i afegeix el nom a la llista d'imatges. El botó *continua* acciona la funció *crearacciotextimatge*(), que a l'igual que en els anteriors constructors, serveix per crear el fitxer de text corresponent a la prova que s'està organitzant. El fitxer té la següent estructura:

(1) Primera línia: paraula "TEXT+IMATGE". Aquesta paraula permetrà identificar el tipus d'acció quan l'alumne connecti amb el servidor.

(2) Segona línia: nom de la imatge del material: Serà necessari per carregar el material des del servidor.

(3) Tercera línia: Pregunta introduïda al camp d'entrada.

(4) Següents n línies (sent n el número de targets): Noms de les imatges seleccionades.

(5) Següents n línies (sent n el número de targets): Respostes.

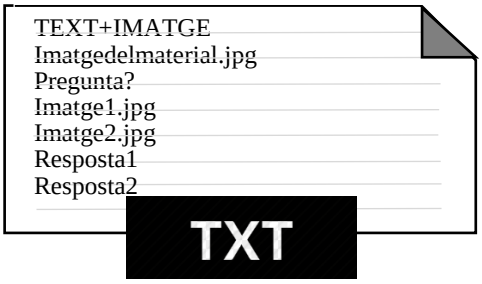

Fig. 4.13. Model fitxer acció text+imatge

A la **Fig 4.14** es pot veure el panell constructor text + imatge.

Fig. 4.14: Panell text + imatge.

4.2.1.3. Constructor de formulari.

El constructor de la prova de tipus formulari s'assembla molt al constructor de l'acció de text. He introduït un camp d'entrada on l'administrador ha d'escriure el link al formulari que l'estudiant ha de contestar el dia de la gimcana.

Fig. 4.15. Constructor formulari.

El botó *continua* acciona la funció *crearaccioformulari(),* que a l'igual que en els anteriors constructors, serveix per crear el fitxer de text corresponent a la prova que s'està organitzant. El fitxer té la següent estructura:

(1) Primera línia: paraula "FORMULARI". Aquesta paraula permetrà identificar el tipus d'acció quan l'alumne connecti amb el servidor.

(2) Segona línia: nom de la imatge del material: Serà necessari per carregar el material des del servidor.

(3) Tercera línia: link del formulari introduït al camp d'entrada.

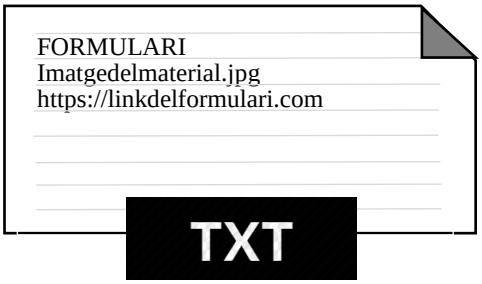

Fig. 4.16: Model fitxer formulari.

4.2.1.4. Constructor de Trivial.

Pel disseny d'aquest constructor he decidit utilitzar una scroll view on s'afegeixen automàticament els camps necessaris per organitzar les preguntes del trivial. Apareixeran tantes preguntes per omplir com targets hagi escollit l'administrador a les dades generals de l'activitat. A la **Fig 4.17** es pot veure l'estructura d'aquest constructor. En la primera casella apareix el número de pregunta, en la segona un camp per escriure la pregunta desitjada, les quatre següents són per omplir les diferents respostes, i la cinquena és un desplegable per escollir quina resposta és la correcta (a,b,c,d).

El codi responsable d'aquest sistema es troba a la funció *seguentaccio()*, que consulta el número de targets a la *clasegenerica* i apartir d'aqui instancia els elements necessaris d'una sèrie d'objectes base (*plantilles*) per omplir la scroll view.

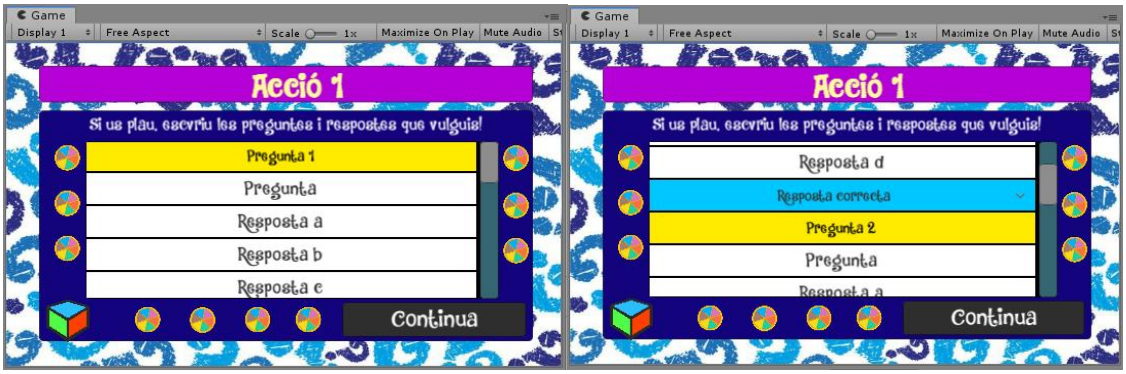

Fig. 4.17. Constructor del trivial.

El botó *continua* acciona la funció *crearacciotrivial(),* que al igual que en els anteriors constructors, serveix per crear el fitxer de text corresponent a la prova que s'està organitzant.

El fitxer té la següent estructura:

(1) Primera línia: paraula "TRIVIAL". Aquesta paraula permetrà identificar el tipus d'acció quan l'alumne connecti amb el servidor.

(2) Segona línia: nom de la imatge del material: Serà necessari per carregar el material des del servidor.

(3) Tercera línia, novena línia: Pregunta 1, Pregunta 2...

(4) Quarta línia, desena línia: Resposta A1, Resposta A2...

(5) Cinquena línia, onzena línia: Resposta B1, Resposta B2...

(6) Sisena línia, dotzena línia: Resposta C1, Resposta C2...

(7) Setena línia, tretzena línia: Resposta D1, Resposta D2...

(8) Setena línia, catorzena línia: Resposta correcta 1, Resposta correcta 2 ...

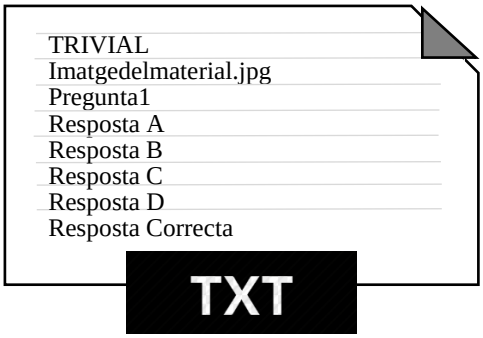

Fig. 4.18. Model fitxer trivial

Finalment, he creat un panell de retorn al menú principal que s'activa quan totes les proves ja s'han organitzat. Conté un missatge i un botó per retornar al menú principal de l'aplicació. A més he afegit un recordatori del codi que hauran de posar els alumnes per accedir a les activitats. Aquest panell es pot veure a la **Fig 4.19.**

Fig. 4.19: Panell de retorn.

4.2.2. Pujada de fitxers directa.

Aquesta opció és útil per tots aquells administradors que ja saben com funciona la dinàmica de l'aplicació, i permet carregar els fitxers de text i imatges d'una activitat al servidor directament. Per poder utilitzar aquesta opció s'ha de saber prèviament com és l'estructura dels fitxers per les diferents proves (en l'anterior apartat està especificat).

Per aquesta proposta he dissenyat un únic panell que a través de dos drop downs permet pujar tots els fitxers. Un serveix per carregar fitxers de text, i l'altre per carregar imatges. Per fer-ho he afegit dos botons que accionen les funcions *loadfile() i loadimage*() respectivament, i que obren un quadre de diàleg per seleccionar els arxius. Una vegada seleccionats es manen al servidor i apareixen els noms a sota dels botons. A la **Fig 4.20** es pot veure l'aparença d'aquest panell.

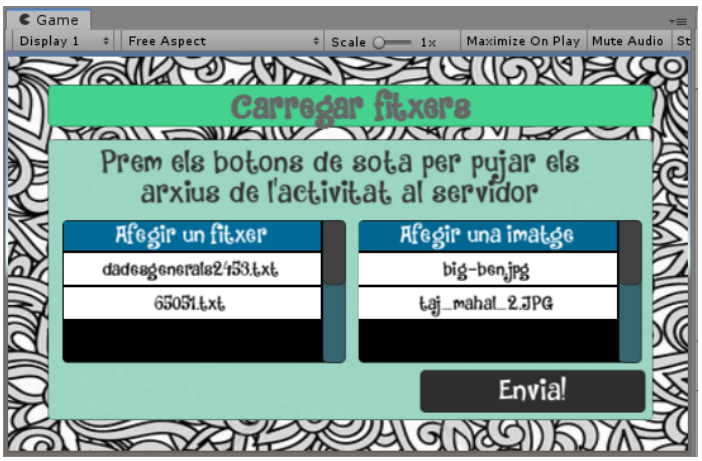

Fig. 4.20. Panell per carregar fitxers.

4.3. Escena de l'alumne.

L'escena dels alumnes serveix per jugar i accedir a les activitats generades pels administradors. Tal i com hem vist en l'anterior secció els administradors introdueixen un codi que identifica cadascuna de les partides generades, i que han de facilitar als alumnes per poder jugar.

4.3.1. Panell de registre.

Partint d'aquí, el primer panell que he afegit en aquesta escena és un de registre. En aquest panell els grups d'alumnes han d'especificar l'escola a la que pertanyen, quin número de grup són, i el codi identificador de l'activitat. Quan els alumnes premen el botó *comença* succeeix el següent:

- S'executa la funció *carregarjoc().* Aquesta funció és una de les més importants de l'aplicació, si no la que més, i s'encarrega bàsicament de carregar l'activitat.

- Es tanca el panell per donar lloc a la partida.

Fig. 4.21. Panell de registre.

Per poder carregar el joc, abans hem de crear tots els elements necessaris per carregar-lo.

4.3.2. Image Targets.

El primer element clau és una base de dades de targets que prèviament he creat al target manager de Vuforia. Una vegada importada la base de dades a Unity, he creat 20 objectes del tipus imagetarget[7] i els he agrupat en un de sol anomenat *targets*. Els he anomenat igual que el seu Index (1,2,3...) per facilitar-ne el codi i els he adjudicat a cadascú la seva corresponent imatge de la base de dades. A la **Fig 4.22** es poden apreciar tots els targets.

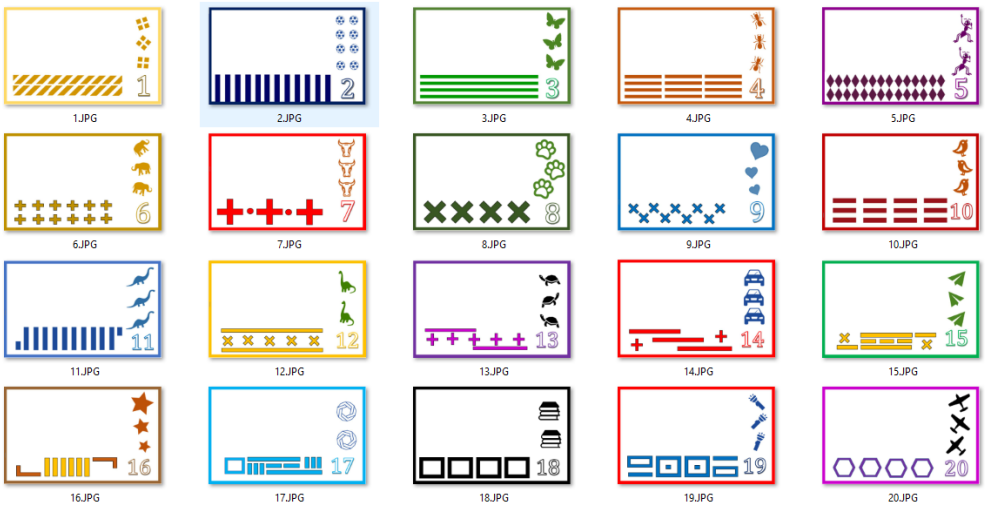

Fig. 4.22. Col·lecció de targets.

[7] Els imagestargets són objectes que contenen la informació dels targets.

4.3.3. Cubs.

Una vegada que ja tenim els targets creats és hora de crear els cubs que contindran la informació de les proves. Per aquesta proposta, he creat una carpeta local anomenada *cubs*, en la qual he afegit quatre prefabs: *cubtext, cubtextimatge, formulari i trivial*. En cadascun d'aquests prefabs[8] he afegit un component d'script per gestionar la informació que contenen.

- **Cubtext:** És el cub que conté la informació de la prova de tipus text. Per introduir la informació he optat per afegir un una variable pública anomenada *inputtascacontructor0* a l'script. Aquesta variable és de tipus text i conté el missatge de la prova que prèviament ha programat l'administrador.

- **Cubtextimatge**: És el cub que conté la informació de la prova de tipus text+imatge. Per introduir la informació he optat per afegir tres variables públiques anomenades *inputtascacontructor1, imagepath i resposta*. Aquestes variables són de tipus text i contenen la pregunta comú, el nom de la imatge corresponent al target (tots els targets tenen una imatge diferent) actual, i la resposta a la pregunta.

- **Formulari:** És el cub que conté la informació de la prova de tipus formulari. Per introduir la informació he optat per afegir un una variable pública anomenada *linkdelformulari* a l'script. Aquesta variable és de tipus text i conté el link del formulari de la prova. Aquesta variable s'emplena en el moment de carregar el joc quan el jugador clica a comença.

- **Trivial:** És el cub que conté la informació de la prova de tipus trivial. L'script conté 5 variables públiques de tipus text. Una què es la pregunta, quatre corresponents a les respostes a,b,c i d, i finalment una altre per indicar quina resposta es la correcta.

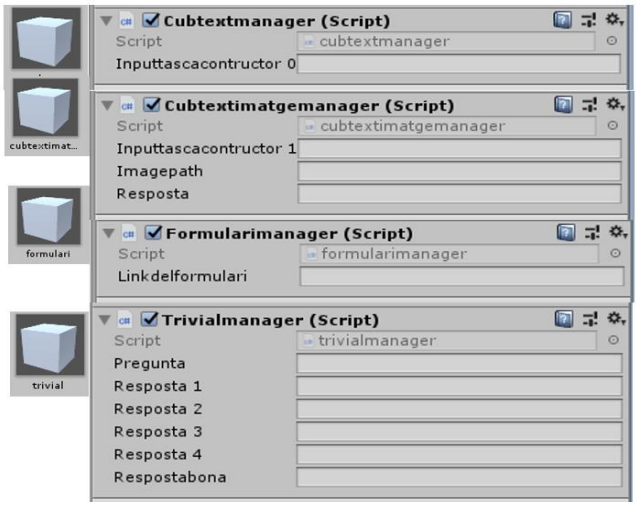

Fig. 4.23. Scripts dels prefabs.

[8] Els prefabs són objectes guardats en local amb tots els seus components, propietats i objectes secundaris, que actuen com una plantilla des de la que es poden crear noves instancies prefabricades a l'escena.

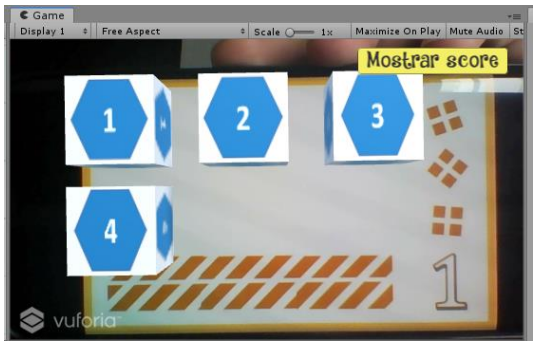

Fig. 4.24. Vista dels cubs sobre un target.

Seguidament he afegit el mètode *OnMouseDown()* a cadascun dels scripts, que primer comprova quina és la distancia entre la localització del smartphone i l'establerta pel target. Si l'alumne no es troba dins el radi de proximitat, s'acciona la funció *activarpanellavis()* del *gamemanag* amb el missatge: "encara no has arribat a la localització".

Seguidament consulta a la classe genèrica si s'han de fer les accions en ordre, i si s'ha de fer només una o totes. Si només s'ha de fer una acció, es comprova que aquella acció no s'hagi executat en una altre localització, o si ja s'ha realitzat alguna prova en la mateixa localització. En cas afirmatiu s'acciona el panell d'avís amb els missatges "aquesta acció ja s'ha realitzat en una altre localització" i "ja has fet una acció d'aquest target" respectivament.

Si les localitzacions s'han de visitar en ordre comprova que les proves de l'anterior target hagin estat completades. En cas negatiu s'acciona el panell d'avís amb el missatge "Primer has de completar els anteriors targets".

Fig. 4.25. Panells d'avís.

Si cap de les condicions anteriors és compleix, llavors la proba es desbloqueja.
I com es manifesten les proves? A excepció de l'acció de formulari, que simplement s'ha d'obrir el link des del navegador per omplir el formulari, les altres tres proves es manifesten amb panells que s'emplenen amb la informació que contenen els cubs.

4.3.4. Panells de les proves.

4.3.4.1. Panell acció text.

Aquest panell té un quadre de text on es mostra el missatge a realitzar pels participants de la gimcana, i un botó per tancar el panell. S'activa amb la funció *activarpanelltext(),* que té com a entrada el missatge, el numero de target, i el numero d'acció.

Fig. 4.26. Panell de text.

4.3.4.2. Panell acció text+imatge.

Aquest panell té un quadre text on es mostra la pregunta corresponent a la localització on s'estiguin fent les proves, la corresponent imatge, i un camp d'entrada per introduir la resposta. S'activa amb la funció *activarpanelltextimatge*(), que té com a entrada la pregunta, el nom de la imatge a carregar, la resposta correcta, el numero de target i el número d'acció.

Fig. 4.27. Panell de imatge+text.

4.3.4.3. Panell trivial.

Aquest panell té un quadre de text on es mostra la pregunta del trivial corresponent a la localització on s'estiguin fent les proves i quatre botons per seleccionar la resposta. S'activa amb la funció *activartrivial()*, que al igual que els altres panells té com a entrada els elements a carregar en el panell.

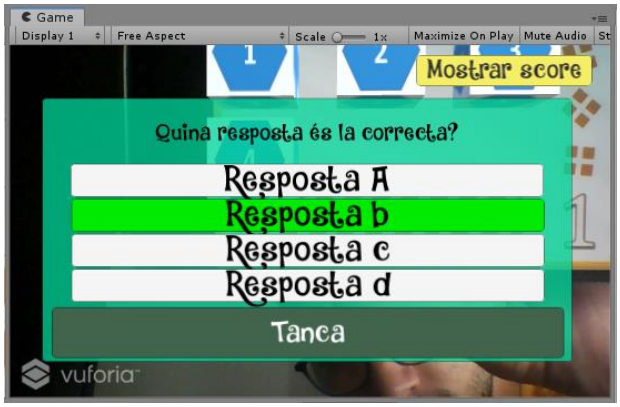

Fig. 4.28. Panell trivial.

Cal esmentar que totes les funcions per activar un panell s'executen cada vegada que polsem sobre un cub si es compleixen les condicions que hem explicat en l'anterior apartat. Per tenir un control de les accions que s'han fet i les que no, he afegit a la classe genèrica un array bidimensional de dimensions *ntargets x nproves* que inicialment s'emplena tot de zeros. Cada vegada que es fa una acció s'actualitza la matriu amb un 1, un 2 o un -2. Un 1 significa que l'acció s'ha realitzat. Un 2, significa que l'acció era una pregunta i s'ha respòs bé, i un -2, que s'ha respòs malament.

4.3.4.4. Panell score.

Aquest panell permet a l'usuari mantenir-se informat de quines proves ha realitzat i quines no. Podríem dir que és un element de feedback per l'alumne, i per tant ha de ser fàcil de consultar i visual. Per aquest motiu he optat per dissenyar una graella, on les files siguin les diferents localitzacions, i les columnes les diferents proves. Per aquesta proposta he afegit una scroll view al panell que s'emplena a partir d'una funció anomenada *setgridscores().*

Aquesta funció consulta totes les dades necessàries de la clasegenerica per omplir el grid. Com el número de columnes varia segons el número de proves per target, la graella té dimensions variables en cada cas i per tant si fixem una mida de les caselles constant no obtindríem un resultat adequat. Per resoldre aquest problema he optat per definir la mida de les caselles en funció del número de columnes que hagi de tenir. Si hi ha més columnes, les caselles seran més petites, i si ha menys, seran més gran. D'aquesta manera aconseguim que les dades no es desquadrin i que sempre una fila correspongui a un target, i una columna a un tipus de prova. A continuació es mostra una captura d'aquest panell en una activitat formada per 6 targets i 4 proves.

Adicionalment, en aquest panell he incorporat un botó que s'encarrega de manar els resultats a la base de dades i que els alumnes han de polsar en acabar l'activitat. La funció que s'encarrega d'això es diu *manarresultats().*La taula de la base de dades està formada per tres columnes: El nom de l'escola, el número de grup i la puntuació. Les tres variables estan emmagatzemades a la *clasegenerica* així que només s'ha de consultar el valor i enviar-lo.

Fig. 4.29. Panell score.

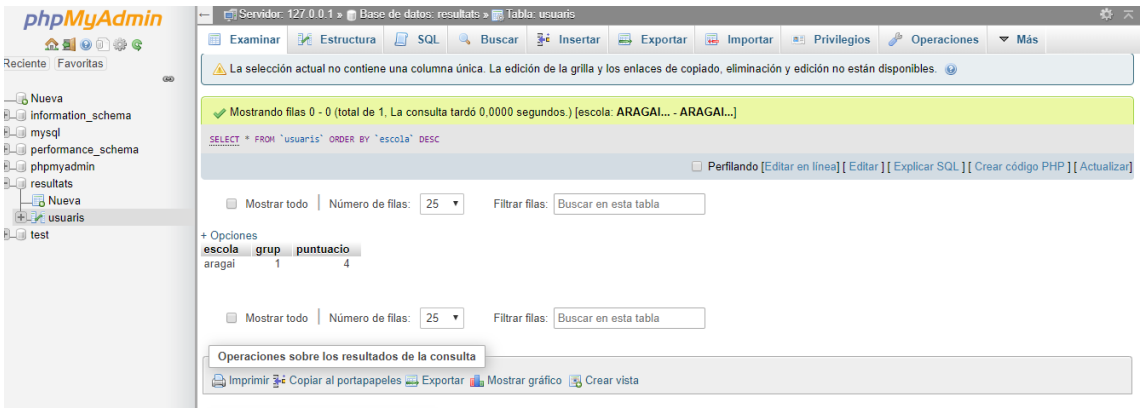

Fig. 4.30. Exemple d'un usuari a la base de dades.

4.3.5. Càrrega de l'activitat.

Tal i com hem dit l'activitat es carrega quan l'alumne ha introduït les dades al panell de registre i ha polsat el botó comença, executant la funció *carregarjoc().* Aquesta funció segueix els següents pasos:

(1) Es connecta amb el servidor per descarregar el fitxer amb les dades generals de l'activitat. Llegeix el fitxer i guarda les dades en variables.

(2) Crea una matriu de zeros (ntargets x nproves) targets i la mana a la classe genèrica.

(3) Crea una matriu que conté les coordenades dels targets i la mana a la classe genèrica.

(4) Activa tants targets com s'indica al fitxer de dadesgenerals.

(5) Executa un bucle dins d'un altre bucle. En l'exterior recorre el número de proves, i en l'interior el número de targets. En definitiva descarrega els fitxers de les accions un per un i comprova de quin tipus d'acció es tracten llegint-ne la primera línia. Llavors s'instancien i es generen copies dels cubs a cadascun dels targets, tot introduint la informació que han de contenir.

(6) Es tanca el panell de registre i s'activa el botó *mostrarscore*. Aquest botó activa un panell que mostra les proves realitzades en una graella.

Capítol 5. Flexibilitat de l'aplicació.

Tal i com vam definir a la introducció i a les especificacions de l'aplicació, un dels requeriments d'aquest projecte era aconseguir una aplicació que, lluny de ser una versió definitiva, en permetés realitzar canvis sobre aquesta d'una forma senzilla. Per tal de cobrir aquest objectiu hem de tenir diversos factors en compte alhora de dissenyar l'aplicació:

- Que el codi sigui fàcil d'interpretar i que incorpori explicacions del que s'està fent en tot moment.
- Que els diferents objectes i escenes estiguin ben organitzats dins de la finestra de jerarquies.
- Que el programador que vulgui treballar sobre la nostre aplicació tingui alguna mena d'ajuda per introduir noves funcionalitats.

5.1. Interpretació del codi.

El primer que hem de saber és que tots els scripts es troben dins la carpeta local /scripts que a la vegada e troba dins de /Assets, tal i com es pot veure a la **Fig 5.1.**

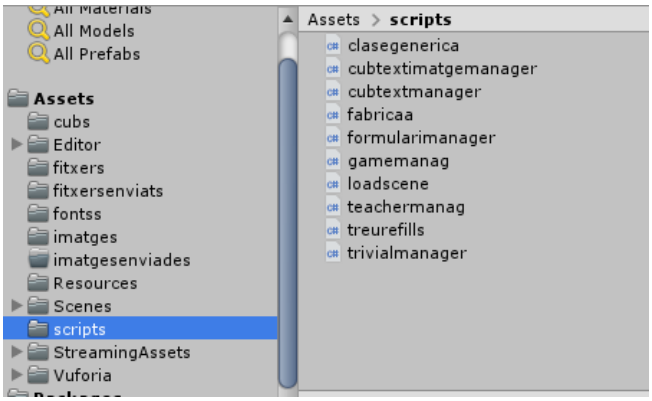

Fig. 5.1. carpeta scripts.

Tots els scripts incorporen comentaris que expliquen el codi per passos per tal que sigui fàcil d'interpretar: Hi ha comentaris al costat de les variables definides, abans dels mètodes (explicant perquè serveixen), i a dins dels mètodes (explicant com ho fan). Al costat del nom de la classe també s'incorporen comentaris tot explicant per què serveix la classe en si.

Tots els scripts es poden trobar als annexos d'aquests document, però a la **Fig 5.2, 5.3, 5.4 i 5.5** es facilita una mostra de com estan estructurats els codis.

```
public class teachermanag : MonoBehaviour //aquesta classe gestiona les accions dutes a terme pels administradors.
{
```

Fig. 5.2. Explicació de la classe.

```
// ------------------------------------------------------------------------------------------------------------------------
// ----------------------------------------------------VARIABLES-----------------------------------------------------------
// ------------------------------------------------------------------------------------------------------------------------

//primer declarem els dfiferents panells utilitzats per generar les partides
public GameObject panelltext; // Panell constructor d' accions de text.
public GameObject panelltextimatge; //Panell constructor d' accions de tex + imatge.
public GameObject panellformulari; //Panell constructor d' accions formulari.
public GameObject panelltrivial; //Panell constructor d' accions de trivial.
public GameObject tipusaccions; // panell en el que es decideixen el tipus de proves de la gimcana.
public GameObject panellfitxers; // panell per pujar fitxers al servidor directament.
public GameObject panelldadesgen; //panell on s'introdueixen les dades generals de la gimcana.
public GameObject escollirmode; // panell d'elecció entre generar una partida manualment o pujar fitxers al server directament.
public GameObject panellfinal; // última finestra que s'bre quan ja s'ha generat l'activitat.
public GameObject panellcoordenades;


// ------------------------------------------------------------------------------------------------------------------------
// ----------------------------------------------------FUNCIONS-----------------------------------------------------------
// ------------------------------------------------------------------------------------------------------------------------
// la funció introduir dadesgenerals() s'encarrega de generar un fitxer de text amb les dades generals de l'activitat.
// Llavors truca a la corutina dadgenerals() que mana el fitxer al servidor.
public void introduirgenerals()...
// la corutina dadgenerals() mana el fitxer de dades generals al servidor: recordem que el fitxer de dades generals rep el nom:
// XXXXdadesgenerals.txt
public IEnumerator dadgenerals()...
//la funció carregar material obre un quadre de diàleg per seleccionar una foto que servirà com a textura pel material del cub
// en el que estem treballant.
public void carregarmaterial()...
// la corrutina carregarmateriall s'encarrega de manar la imatge seleccionada en la funció carregarmaterial() al servidor ja que quan
// l'alumne es vulgui baixar l'activitat totes les imatges les carregarà del servidor.
public IEnumerator carregarmateriall(string pat)...
//la funció settipusaccions consulta al panell "tipusaccions" quin tipus de prova ha asignat l'administrador a cada cub. Recordem que
// aquesta elecció es fa a través de dropdowns. LLavors consultem el valor que té el dropdown i l'afegim a la llista llistaccions de la
// clasegenerica. D'aquesta manera obtenim una llista en la que emmagatzem quins tipus de accions ha seleccionat l'administrador
```

Fig. 5.3. comentaris de variables i mètodes.

```
// la funció introduir dadesgenerals() s'encarrega de generar un fitxer de text amb les dades generals de l'activitat.
// Llavors truca a la corutina dadgenerals() que mana el fitxer al servidor.
public void introduirgenerals()
{
    int i = 0;
    //primer definirem una llista d'strings on guardarem tota la informació seleccionada als dropdowns del panell
    List<string> llistageneral = new List<string>();
    //acontinuació anem afegint aquesta informació a la llista.
    int s = GameObject.Find("/teachercanvas/dadesgen/paneldrops/ntargg").GetComponent<Dropdown>().value + 1;
    int t=  GameObject.Find("/teachercanvas/dadesgen/paneldrops/ncubss").GetComponent<Dropdown>().value + 1;
    llistageneral.Add(s.ToString());
    llistageneral.Add(t.ToString());
    llistageneral.Add((GameObject.Find("/teachercanvas/dadesgen/paneldrops/unaotot").GetComponent<Dropdown>().value).ToString());
    llistageneral.Add((GameObject.Find("/teachercanvas/dadesgen/paneldrops/ordreono").GetComponent<Dropdown>().value).ToString());
    int contador = 0;

    while (contador<s)
    {
        llistageneral.Add(clasegenerica.GetInstance().matriucoordenades[contador, 0].ToString("0.00000", System.Globalization.CultureInfo.InvariantCulture) +" "
        contador = contador + 1;
    }
    //a les següents quatre linies manarem les dades generals a la clase generica:
    //codi identificatiu de la gimcana
    clasegenerica.GetInstance().codicreador = GameObject.Find("/teachercanvas/dadesgen/paneldrops/inpcodi").GetComponent<InputField>().text;
    //numero de targets
    clasegenerica.GetInstance().ntargcreador = s;
    // numero de proves per target
    clasegenerica.GetInstance().naccionscreador = t;
    // binari que defineix si s'ha de fer només una prova o totes
    clasegenerica.GetInstance().unaototescreador = GameObject.Find("/teachercanvas/dadesgen/paneldrops/unaotot").GetComponent<Dropdown>().value;
    // binari que defineix si els targets s'han de visitar en ordre
    clasegenerica.GetInstance().ordrecreador = GameObject.Find("/teachercanvas/dadesgen/paneldrops/ordreono").GetComponent<Dropdown>().value;
    // seguidament generem un fitxer de text on introduim les dades de la llista que hem creat previament
    StreamWriter writer = new StreamWriter(@"Assets/fitxersenviats/dadesgenerals.v",GameObject.Find("/teachercanvas/dadesgen/paneldrops/inpcodi").GetComponent<I
```

Fig. 5.4. Explicacions dels mètodes.

5.2. Guia per introduir noves funcionalitats.

Dedicaré aquest apartat a fer una guia que serveixi a futurs programadors per afegir noves funcionalitats a l'aplicació, com ara introduir proves a la gimcana o canviar el número màxim de proves per localització.

5.2.1. Com podria introduir un nou tipus de prova?

Per introduir un nou tipus de prova la primera cosa que hauríem de fer és introduir el corresponent panell constructor a l'escena dels administradors, tal i com es mostra a la **Fig 5.5**.

Fig. 5.5. Panell constructor per introduir un nou tipus de prova.

Com ja hem explicat anteriorment, el botó *continua* de cada panell constructor executa una funció que genera el corresponent fitxer de text i seguidament crida a la funció *manaraccio(fitxer)* per tal d'enviar el fitxer al servidor. SI volguéssim introduir un nou tipus de prova, després de crear el panell constructor hauríem d'escriure la corresponent funció per generar els fitxers. A la **Fig 5.6** es pot veure una plantilla que he dissenyat per una futura prova.

```
//la funció crearnoutipusdeprova() serviría per generar el fitxer de la prova i manar-lo al servidor
public void crearnoutipusdeprova()
{
    //Primer definim una llista que contindrà tota la informació per després emplenar el fitxer
    List<string> dadesnovaaccio = new List<string>();
    dadesnovaaccio.Add("NOU TIPUS DE ACCIÓ"); //capçalera del fitxer
    dadesnovaaccio.Add(Path.GetFileName(clasegenerica.GetInstance().materialaccio)); //material
    int i = 0;
    // seguidament anem afegint la informació a la llista
    while (i < clasegenerica.GetInstance().ntargcreador)
    {
        dadesnovaaccio.Add("ANAR AFEGINT LA INFORMACIÓ DE LA PROVA");
        i = i + 1;
    }
    //seguidament generem el fitxer de text
    StreamWriter arxiudadesnovaprova = new StreamWriter(@"Assets/fitxersenviats/" + (clasegenerica.GetInstance().naccionscreador -
    int jjj = 0;
    //i anem afegint les dades de la llista
    while (jjj < dadestrivial.Count)
    {
        arxiudadesnovaprova.WriteLine(dadesnovaaccio[jjj]);
        jjj = jjj + 1;
    }
    arxiudadestrivial.Close();
    StartCoroutine(manaraccio("Assets/fitxersenviats/" + (clasegenerica.GetInstance().naccionscreador - clasegenerica.GetInstance()
}
```

Fig. 5.6. Plantilla per un nou tipus de prova.

Seguidament hauríem de preparar la prova a l'escena de l'alumne. Només hauríem de fer dues coses: Crear un nou prefab pel nou tipus d'acció, i modificar el codi de la funció *carregarjocc()* per incloure també a la nova prova.

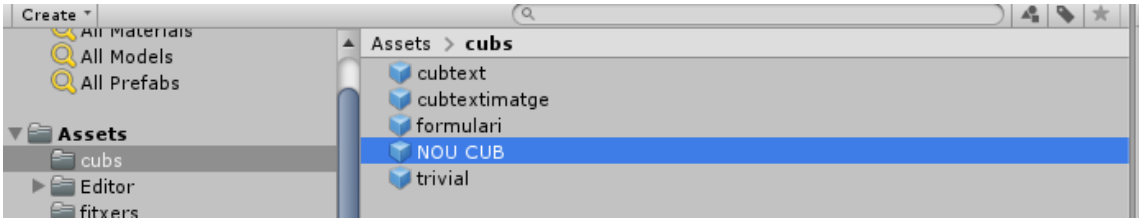

Fig. 5.7. Nou prefab.

```
        }
        else if (accioacarregar[0] == "NOU TIPUS DE ACCIO")
        {
            //Aqui hauriem de introduir el codi per carregar la prova
        }
        else if (accioacarregar[0] == "TEXT+IMATGE") //Acció de tipus text+imatge
```

Fig. 5.8. Modificació de la funció carregarjoc()

I ja tindríem el nou tipus de prova preparat.

5.2.2. Com podria canviar l'aparença dels targets? I afegir-ne més?

Els targets que he dissenyat per la nostra aplicació son de caire generalista, és a dir no contenen informació visual d'una localització en específic, si no que estan pensats per utilitzar-se en qualsevol gimcana. En cas de que es volgués canviar l'aparença dels targets s'hauria de crear una nova base de dades al Vuforia developer portal tal i com s'indica a la **Fig 5.9,** i després importar-la a Unity.

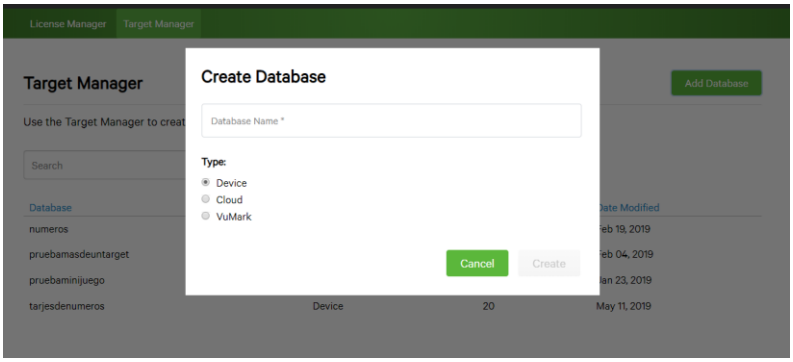

Fig. 5.9. Nova base de dades.

Una futura funcionalitat que estaria bé implementar és tenir la possibilitat de decidir la textura dels imatge targets en el moment de l'execució sense haver de carregar les bases de dades manualment. Això es podria fer amb el servei de hosting de bases de dades que ofereix Vuforia, però s'ha de pagar apart.

Per afegir nous targets s'haurien d'afegir com a fills de l'objecte targets, introduint l'índex corresponent i seleccionant la imatge identificadora de la base de dades importada.

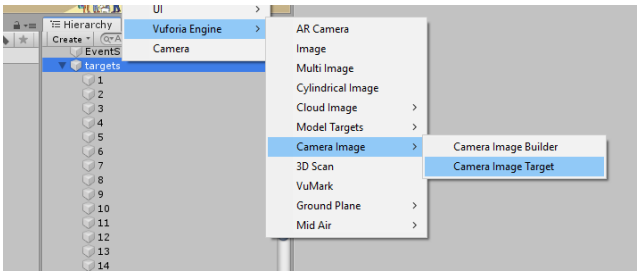

Fig. 5.10 Afegint un nou target.

Capítol 6. Conclusions i línies futures.

6.1. Assoliment d'objectius.

Tal i com vam definir a la introducció, l'objectiu principal d'aquest projecte era aconseguir desenvolupar una aplicació per gestionar les proves de l'activitat proposada pel WMCP. Una aplicació que incorporés un menú d'administradors per generar partides de forma personalitzada sense la necessitat de tenir coneixements de programació, i que a la vegada incorporés una interfície pels estudiants, tal i com ja havia a l'anterior. Una vegada acabat el projecte podem afirmar que hem assolit aquest objectiu.

Hem desenvolupat una aplicació basada en el principi administrador-servidor-usuari, és a dir, l'administrador genera una partida i es puja al servidor en forma de fitxers, que més tard l'alumne es descarrega per poder jugar. Hem incorporat dues modalitats per generar les activitats: de forma manual a través de panells, o pujant directament els fitxers que seleccionem de la unitat d'emmagatzematge. Per altra banda la interfície d'alumne descàrrega els fitxers de text i els llegeix per carregar les proves en temps real. També compta amb un menú de seguiment des del que es poden manar els resultats de les proves. Les eines utilitzades per portar aquest projecte a terme han sigut Unity 3D, Vuforia i XAMPP.

6.2. Impacte.

Aquest projecte és un projecte més que confirma com la tecnologia està canviant el mon de l'educació per tal d'oferir un aprenentatge més dinàmic, on els alumnes aprenen per motivació pròpia i no per obligació. En especial la RA pot ser una bona opció ja que molts dels estudiants atribueixen aquesta tecnologia a videojocs que els agraden. D'aquesta manera veuen l'aprenentatge com un videojoc en el que ells mateixos son els personatges. Per altra banda els docents també veuen l'oportunitat d'expressar les seves tècniques i coneixements en forma d'activitats pels estudiants.

6.3. Línies futures.

Tal i com s'ha comentat en el Capítol 5, aquesta aplicació està pensada per afegir noves funcionalitats en un futur, Algunes de les millores que es podrien incorporar són:

- Afegir nous tipus de proves a l'activitat.
- Implementar un servidor més robust i crear una pàgina web amb un domini on es puguin penjar els fitxers.
- Crear una espècie de lliga en la que es puguin veure les puntuacions i les proves que han hagut de fer els companys d'altres localitats

6.4. Conclusions personals.

Aquest projecte m'ha servit per reforçar els meus coneixements de programació i introduir-me en el món de Unity 3D. He après que en un projecte en el que hi ha un número considerable de mètodes i variables és molt important tenir les classes ben organitzades i dividides per funcionalitats. També he aprés a implementar un servidor amb XAMPP i fer-lo públic. M'agradaria desenvolupar aplicacions què involucressin més funcionalitats de Unity en un futur mentre milloro les meves habilitats. Finalment comentar que he sentit com aquest projecte pot servir en un futur com a base per afegir noves funcions, la qual cosa em deixa satisfet i amb ganes de continuar.

BIBLIOGRAFIA

[1] Manual d'usuari de Unity 3D, https:// docs.unity3d.com/ es/ curent/ Manual/ index.html.

[2] Fòrums d'Stackoverflow, https://stackoverflow.com/.

[3] Fòrums de Unity 3D, https://answers.unity.com.

[4] Informació de RA a l'educació, https:// www.educaciontrespuntocero.com/ noticias/ usos-realidad-aumentada-aulas/91867.html

[5] Informació de posicionament GPS, https://acolita.com/como-funcionan-los-dispositivos-gps-trilateracion-vs-triangulacion/.

[6] Informació Realitat Augmentada, https:// www.realitytechnologies.com/ augmented-reality/.

[7] Vuforia developer portal, https://developer.vuforia.com/.

[8] YouTube (vídeo tutorials), https://www.youtube.com/

[9] World Mobile City Project web, http://www.wmcproject.org/?lang=es.

[10] Guia de Microsoft, https:// docs.microsoft.com/ es-es/ dotnet/ api /system.math? view=netframework-4.8

[11] Guia didàctica Barcelonada. Document facilitat a la Web del MWCP.

ANNEXOS

gamemanag.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using System.Text;
6. using System;
7. using Vuforia;
8. using System.Net;
9. using UnityEditor;
10.      using UnityEngine.EventSystems;
11.      using UnityEngine.SceneManagement;
12.      using System.IO;
13.      using System.Linq;
14.      using System.Globalization;
15.      public class gamemanag : MonoBehaviour
16.      {
17.          // ------------------------------------------------------
    ------------------------------------------------------------------
    ----------------------------
18.          // ------------------------------------------------------
    VARIABLES-------------------------------------------------------
    ----------------------------
19.          // ------------------------------------------------------
    ------------------------------------------------------------------
    ----------------------------
20.          //Primer que tot declarem les variables necessaties per
    la execució de les funcions de la classe gamemanag.
21.          //(no totes les variables de l'activitat s'emmagatzemen
    en aquesta classe, algunes es troben a la clasegenerica)
22.          public GameObject codi; // el codi de l'activitat
23.          public GameObject panelltext; //el panell per l'acció
    de text.
24.          public GameObject panelltextimatge; //el panell per
    l'accíó de tipus text+imatge
25.          public GameObject panelltrivial; // el panell per
    l'acció de tipus trivial
26.          private int targetseleccionat; // les variables
    targetseleccionat i accioseleccionada
27.          private int accioseleccionada; // s'actualitzen cada
    cop que pulsem sobre un cub.
28.          private string respostacorrectatrivial; //aquesta
    variable emmmagatzema la resposta correcta del trivial que
    s'estigui executant
29.          private int acertonotrivial; // acertonotrivial val 0
    si l'usuari falla la pregunta o 1 si encerta (pregunta del
    trivial)
30.          public GameObject mostrarscor; //el botó per mostrar el
    panell de resultats
31.          private int contadortrivial = 0; //contadortrivial és
    un enter que val 0 si la pregunta de trivial encara no s'ha
    respós, i 1 quan ja s'ha respós
32.          public GameObject gridd; //el panell (graella) de
    resultats
33.          public GameObject nomescola; //nom de l'escola
34.          public GameObject numerogrup; //número de grup
```

```
35.         public InputField inputresposta; // el camp d'entrada
   per la resposta de l'acció text+imatge
36.         public List<WWW> adrsss = new List<WWW>();
37.         public GameObject panellavisos;
38.         public float latitude;
39.         public float longitude;
40.         // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
41.         // ------------------------------------------------------
   --FUNCIONS-------------------------------------------------------
   ----------------------------
42.         // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
43.         //la funció carregarjoc executa la corutina
   carregarjocc()
44.         public void carregarjoc()
45.         {
46.             StartCoroutine(carregarjocc());
47.         }
48.         //La corrutina darregarjoc() s'encarrega de carregar
   l'activitat creada per l'administrador prèviament
49.         public IEnumerator carregarjocc()
50.         {
51.             //el primer que fem és descarregar-nos el fitxer de
   dades generals desde el servidor
52.             //Tal i com es pot veure a la linia 49 es busca
   l'arxiu apartir del codi que ha picat l'alumne prèviament al
   panell de rgistre
53.             WWW
   adrs = new WWW("http://84.77.48.234:8523/miservidor/ficheros/dad
   esgenerals" + codi.GetComponent<Text>().text + ".txt");
54.             yield return adrs;
55.             //una vegada tenim l'arxiu farem una copia a la
   carpeta local anomenada "fitxers"
56.             File.WriteAllBytes(@"Assets/fitxers/dadesgenerals"
   + codi.GetComponent<Text>().text + ".txt", adrs.bytes);
57.             //a continuació generem un vector de strings que
   conté tota la informació del fitxer, i asignem les variables de
   la clasegenerica
58.             // (numero de targets, numero de accions, una o
   totes, ordre o no ordre, i les coordenades dels targets)
59.             string[] vectordadesgenerals = System.IO.File.ReadA
   llLines(@"Assets/fitxers/dadesgenerals"+ codi.GetComponent<Text>
   ().text+".txt");
60.                 clasegenerica.GetInstance().ntarg= int.Parse(ve
   ctordadesgenerals[0]);
61.                 clasegenerica.GetInstance().naccions = int.Pars
   e(vectordadesgenerals[1]);
62.                 clasegenerica.GetInstance().unaototes = int.Par
   se(vectordadesgenerals[2]);
63.                 clasegenerica.GetInstance().ordre = int.Parse(v
   ectordadesgenerals[3]);
64.                 clasegenerica.GetInstance().setmatriucontrolacc
   ions(int.Parse(vectordadesgenerals[0]), int.Parse(vectordadesgen
   erals[1]));
65.             int contcord = 0;
66.             double[,] vectordecoordenades = new double[20, 2];
67.             while (contcord< clasegenerica.GetInstance().ntarg)
68.             {
```

```
69.                 string[] latitudilongitud = vectordadesgenerals
    [contcord+4].Split(' ');
70.                 vectordecoordenades[contcord, 0] = System.Conve
    rt.ToDouble(latitudilongitud[0], CultureInfo.InvariantCulture);
71.                 vectordecoordenades[contcord, 1] = System.Conve
    rt.ToDouble(latitudilongitud[1], CultureInfo.InvariantCulture);
72.                 contcord = contcord + 1;
73.             }
74.             clasegenerica.GetInstance().matriucoordenades = vec
    tordecoordenades;
75.             int cont = GameObject.Find("targets").transform.chi
    ldCount;
76.             int ff = 0;
77.             //a continuació descarregarem del servidor els
    fitxers que contenen la informació de les proves
78.             // Com? primer consultem el numero de proves per
    target, variable que acabem d'obtenir amb el fitxer de dades
    generals
79.             // Llavors fem un loop que va descarregant els
    fitxers fins que l'iterador sigui igual al número de proves.
80.             while (ff< clasegenerica.GetInstance().naccions)
81.             {
82.                 int fff = ff + 1;
83.                 adrsss.Add(new WWW("http://84.77.48.234:8523/mi
    servidor/ficheros/"+ fff + codi.GetComponent<Text>().text + ".tx
    t"));
84.                 yield return adrsss[ff];
85.                 //Al igual que el fitxer de dades generals,
    també guardem els fitxers de les proves a la carpeta "fitxers"
86.                 File.WriteAllBytes(@"Assets/fitxers/"+fff+ codi
    .GetComponent<Text>().text + ".txt", adrsss[ff].bytes);
87.                 ff = ff + 1;
88.             }
89.             // En un principi tots els targets (de l'1 al 20)
    estan activats, però en aquest pas consultarem el número de
    targets i anierm
90.             // desactivant targets fins quedarnos amb els que
    volem (començarem desactivant del target 20).
91.             while (cont>(clasegenerica.GetInstance().ntarg))
92.             {
93.                 GameObject.Find("targets/" + cont).SetActive(fa
    lse);
94.                 cont = cont - 1;
95.             }
96.             int i = 0;
97.             //en el següent pas anirem llegint els fitxers de
    text per construir les diferents proves de l'activitat
98.             while (i<=(clasegenerica.GetInstance().naccions-1))
99.             {
100.                // Al loop exterior anirem recorrent totes les
    proves, i al lopp interior totes els targets per cadascuna de
    les proves
101.                int j = i + 1;
102.                string[] accioacarregar = System.IO.File.ReadAl
    lLines(@"Assets/fitxers/"+ j + codi.GetComponent<Text>().text +
    ".txt");
103.                // el primer que necessitem per carregar una
    prova és saber de quin tipus és. Per això llegirem la primera
    línia del fitxer
104.                // que ens indica quin tipus de prova és "TEXT"
    "TEXT+IMATGE "FORMULARI" o "TRIVIAL"
```

```
105.                 if (accioacarregar[0] == "TEXT") // Acció de
   tipus text.
106.                 {
107.                     //una vegada sabem quin tipus de prova és,
   haurem d'instanciar cubs i omplirlos amb la informació adequada
108.                     // ELs cubs tenen una posició en concret a
   sobre dels targets, tal i com es pot veure a sota.
109.                     //  _____________________________
110.                     // |       |       |       |
111.                     // |   1   |   2   |   3   |
112.                     // |_______|_______|_______|
113.                     // |       |       |       |
114.                     // |   4   |   5   |   6   |
115.                     // |_______|_______|_______|
116.                     //
117.                     // D'això s'encarrega el següent if
118.                     int k = 0;
119.                     string numerodecub = j.ToString(); ;
120.                     // la variable inputt conté el missatge de
   text que llegeix de la linea 3 del fitxer.
121.                     string inputt = accioacarregar[2];
122.                     Vector3 vectorr = new Vector3(0f, 0f, 0f);
123.                     if (numerodecub == "1")
124.                     {
125.                         vectorr = new Vector3(-0.4f, 0f, 0.2f);
126.                     }
127.                     else if (numerodecub == "2")
128.                     {
129.                         vectorr = new Vector3(-0.1f, 0f, 0.2f);
130.                     }
131.                     else if (numerodecub == "3")
132.                     {
133.                         vectorr = new Vector3(0.2f, 0f, 0.2f);
134.                     }
135.                     else if (numerodecub == "4")
136.                     {
137.                         vectorr = new Vector3(-0.4f, 0f, -
   0.05f);
138.                     }
139.                     else if (numerodecub == "5")
140.                     {
141.                         vectorr = new Vector3(-0.1f, 0f, -
   0.05f);
142.                     }
143.                     else
144.                     {
145.                         vectorr = new Vector3(0.2f, 0f, -
   0.05f);
146.                     }
147.                     // aquest és el loopp interior que va
   recorrent tots els targets.
148.                     while (k < clasegenerica.GetInstance().ntar
   g)
149.                     {
150.                         int z = k + 1;
151.                         //la segona linia del fitxer correspon
   al material, així que haurem de descarregar la imatge del
   material desde el servidor
152.                         // per tal de poder copiar la textura i
   agefir-la al cub.
```

```
153.                         WWW
    www = new WWW("http://84.77.48.234:8523/miservidor/imagenes/" +
    accioacarregar[1]);
154.                         yield return www;
155.                         //en la següent linia de codi
    s'instancia un cub de tipus text (tal i com es pot veure aquests
    cubs els podem trobar a la "fabrica")
156.                         GameObject
    objecte = Instantiate(GameObject.Find("/fabrica").GetComponent<f
    abricaa>().cubtext, new Vector3(-1f, 1f, 0f),
    Quaternion.identity);
157.                         //cada cub rebrà com a nom el número de
    prova que contingui
158.                         objecte.transform.name = (numerodecub);
159.                         // L'objecte pare del cub instanciat
    serà el target z
160.                         objecte.transform.parent = GameObject.F
    ind("/targets/" + z).transform;
161.                         // A continuació cambiem la mida per
    defecte i posem una apropiada perque quedi bé sobre el target.
162.                         objecte.transform.localScale = new Vect
    or3(-0.2f, -0.2f, 0.2f);
163.                         // introduim també la posició que ha de
    tenir el cub sobre el target, la qual hem obtingut al if
    anterior.
164.                         objecte.transform.localPosition = vecto
    rr;
165.                         // A continuació cambiem la textura del
    material del cub per defecte i afegim la de la imatge
    descarregada
166.                         objecte.GetComponent<Renderer>().materi
    al.mainTexture = www.texture;
167.                         // Finalment introduim el missatge de
    text que veurà l'alumne quan obri la prova.
168.                         objecte.GetComponent<cubtextmanager>().
    inputtascacontructor0 = inputt;
169.                         k = k + 1;
170.                     }
171.                 }
172.                 else if (accioacarregar[0] == "TEXT+IMATGE") //
    Acció de tipus text+imatge
173.                 {
174.                     // al igual que en la acció de text,
    consultarem el número de prova per ubicar el cub en un lloc o
    altre
175.                     int k = 0;
176.                     string numerodecub = j.ToString() ;
177.                     string inputt = accioacarregar[1];
178.                     Vector3 vectorr = new Vector3(0f, 0f, 0f);
179.                     if (numerodecub == "1")
180.                     {
181.                         vectorr = new Vector3(-0.4f, 0f, 0.2f);
182.                     }
183.                     else if (numerodecub == "2")
184.                     {
185.                         vectorr = new Vector3(-0.1f, 0f, 0.2f);
186.                     }
187.                     else if (numerodecub == "3")
188.                     {
189.                         vectorr = new Vector3(0.2f, 0f, 0.2f);
190.                     }
```

```
191.                        else if (numerodecub == "4")
192.                        {
193.                            vectorr = new Vector3(-0.4f, 0f, -
    0.05f);
194.                        }
195.                        else if (numerodecub == "5")
196.                        {
197.                            vectorr = new Vector3(-0.1f, 0f, -
    0.05f);
198.                        }
199.                        else
200.                        {
201.                            vectorr = new Vector3(0.2f, 0f, -
    0.05f);
202.                        }
203.                        while (k < clasegenerica.GetInstance().ntar
    g)
204.                        {
205.                            int z = k + 1;
206.                            WWW
    www = new WWW("http://84.77.48.234:8523/miservidor/imagenes/" +
    accioacarregar[1]);
207.                            yield return www;
208.                            GameObject
    objecte = Instantiate(GameObject.Find("/fabrica").GetComponent<f
    abricaa>().cubtextimatge, new Vector3(-1f, 1f, 0f),
    Quaternion.identity);
209.                            objecte.transform.name = (numerodecub);
210.                            objecte.transform.parent = GameObject.F
    ind("/targets/" + z).transform;
211.                            objecte.transform.localScale = new Vect
    or3(-0.2f, -0.2f, 0.2f);
212.                            objecte.transform.localPosition = vecto
    rr;
213.                            objecte.GetComponent<Renderer>().materi
    al.mainTexture = www.texture;
214.                            // en les següents tres linies
    introduirem dins de cada cub el nom de la imatge a carregar, la
    pregunta i la resposta
215.                            objecte.GetComponent<cubtextimatgemanag
    er>().inputtascacontructor1 = accioacarregar[2];
216.                            objecte.GetComponent<cubtextimatgemanag
    er>().imagepath = accioacarregar[k + 3];
217.                            objecte.GetComponent<cubtextimatgemanag
    er>().resposta = accioacarregar[k+3+ int.Parse(vectordadesgenera
    ls[0])];
218.                            k = k + 1;
219.                        }
220.                    }
221.                    else if (accioacarregar[0] == "FORMULARI") //ac
    cio de formulari
222.                    {
223.                        // al igual que en les accions anteriors,
    consultarem el número de prova per ubicar el cub en un lloc o
    altre
224.                        int k = 0;
225.                        string numerodecub = j.ToString();
226.                        string inputt = accioacarregar[2];
227.                        Vector3 vectorr = new Vector3(0f, 0f, 0f);
228.                        if (numerodecub == "1")
229.                        {
```

```
230.                         vectorr = new Vector3(-0.4f, 0f, 0.2f);
231.                     }
232.                     else if (numerodecub == "2")
233.                     {
234.                         vectorr = new Vector3(-0.1f, 0f, 0.2f);
235.                     }
236.                     else if (numerodecub == "3")
237.                     {
238.                         vectorr = new Vector3(0.2f, 0f, 0.2f);
239.                     }
240.                     else if (numerodecub == "4")
241.                     {
242.                         vectorr = new Vector3(-0.4f, 0f, -
    0.05f);
243.                     }
244.                     else if (numerodecub == "5")
245.                     {
246.                         vectorr = new Vector3(-0.1f, 0f, -
    0.05f);
247.                     }
248.                     else
249.                     {
250.                         vectorr = new Vector3(0.2f, 0f, -
    0.05f);
251.                     }
252.                     while (k < clasegenerica.GetInstance().ntar
    g)
253.                     {
254.                         int z = k + 1;
255.                         WWW
    www = new WWW("http://84.77.48.234:8523/miservidor/imagenes/" +
    accioacarregar[1]);
256.                         yield return www;
257.                         GameObject
    objecte = Instantiate(GameObject.Find("/fabrica").GetComponent<f
    abricaa>().formulari, new Vector3(-1f, 1f, 0f),
    Quaternion.identity);
258.                         objecte.transform.name = (numerodecub);
259.                         objecte.transform.parent = GameObject.F
    ind("/targets/" + z).transform;
260.                         objecte.transform.localScale = new Vect
    or3(-0.2f, -0.2f, 0.2f);
261.                         objecte.transform.localPosition = vecto
    rr;
262.                         objecte.GetComponent<Renderer>().materi
    al.mainTexture = www.texture;
263.                         // en la següent linia introduirem al
    cub el link al formulari que l'alumne haurà de contestar
264.                         objecte.GetComponent<formularimanager>(
    ).linkdelformulari = accioacarregar[2];
265.                         k = k + 1;
266.                     }
267.                 }
268.                 else if (accioacarregar[0] == "TRIVIAL") //acci
    ó de trivial
269.                 {
270.                     // al igual que en la acció de text,
    consultarem el número de prova per ubicar el cub en un lloc o
    altre
271.                     int k = 0;
272.                     string numerodecub = j.ToString();
```

```
273.                     Vector3 vectorr = new Vector3(0f, 0f, 0f);
274.                     if (numerodecub == "1")
275.                     {
276.                         vectorr = new Vector3(-0.4f, 0f, 0.2f);
277.                     }
278.                     else if (numerodecub == "2")
279.                     {
280.                         vectorr = new Vector3(-0.1f, 0f, 0.2f);
281.                     }
282.                     else if (numerodecub == "3")
283.                     {
284.                         vectorr = new Vector3(0.2f, 0f, 0.2f);
285.                     }
286.                     else if (numerodecub == "4")
287.                     {
288.                         vectorr = new Vector3(-0.4f, 0f, -
    0.05f);
289.                     }
290.                     else if (numerodecub == "5")
291.                     {
292.                         vectorr = new Vector3(-0.1f, 0f, -
    0.05f);
293.                     }
294.                     else
295.                     {
296.                         vectorr = new Vector3(0.2f, 0f, -
    0.05f);
297.                     }
298.                     while (k < clasegenerica.GetInstance().ntar
    g)
299.                     {
300.                         int z = k + 1;
301.                         WWW
    www = new WWW("http://84.77.48.234:8523/miservidor/imagenes/" +
    accioacarregar[1]);
302.                         yield return www;
303.                         GameObject
    objecte = Instantiate(GameObject.Find("/fabrica").GetComponent<f
    abricaa>().trivial, new Vector3(-1f, 1f, 0f),
    Quaternion.identity);
304.                         objecte.transform.name = (numerodecub);
305.                         objecte.transform.parent = GameObject.F
    ind("/targets/" + z).transform;
306.                         objecte.transform.localScale = new Vect
    or3(-0.2f, -0.2f, 0.2f);
307.                         objecte.transform.localPosition = vecto
    rr;
308.                         objecte.GetComponent<Renderer>().materi
    al.mainTexture = www.texture;
309.                         //en les següents linies introduirem la
    pregunta, les 4 respostes possibles i la resposta bona del
    corresponent target
310.                         objecte.GetComponent<trivialmanager>().
    pregunta = accioacarregar[k * 6 + 2];
311.                         objecte.GetComponent<trivialmanager>().
    resposta1 = accioacarregar[k * 6 + 1 + 2];
312.                         objecte.GetComponent<trivialmanager>().
    resposta2 = accioacarregar[k * 6 + 2 + 2];
313.                         objecte.GetComponent<trivialmanager>().
    resposta3 = accioacarregar[k * 6 + 3 + 2];
```

```
314.                         objecte.GetComponent<trivialmanager>().
   resposta4 = accioacarregar[k * 6 + 4 + 2];
315.                         objecte.GetComponent<trivialmanager>().
   respostabona = accioacarregar[k * 6 + 5 + 2];
316.                         k = k + 1;
317.                     }
318.                 }
319.                 i = i + 1;
320.             }
321.             clasegenerica.GetInstance().nomescola = nomescola.G
   etComponent<Text>().text;
322.             clasegenerica.GetInstance().numerogrup = numerogrup
   .GetComponent<Dropdown>().value+1;
323.         }
324.         //la funció tancarpanell() tanca el panell de registre
   de l'alumne i activa el botó per consultar la puntuació
325.         public void tancarpanell()
326.         {
327.             GameObject.Find("canvasalumne/Panel").SetActive(fal
   se);
328.             mostrarscor.SetActive(true);
329.         }
330.         //ka funció tancaroanellscore tanca el panell de la
   puntació
331.         public void tancarpanellscore()
332.         {
333.             gridd.SetActive(false);
334.             mostrarscor.SetActive(true);
335.         }
336.         //La funxió activarpanell text (string textt,int
   targett, int accioo)s'encarrega de activar el panell de text amb
337.         //la informació del cub des del qual s'ha executat
338.         public void activarpanelltext(string textt,int targett,
    int accioo)
339.         {
340.             //tal i com es pot veure la entrada consta del
   missatge a mostrar, el número de target i prova des de el qual
   es crida
341.             //a la funció
342.             panelltext.transform.GetChild(1).gameObject.GetComp
   onent<Text>().text=textt;
343.             //les següents dues variables serveixen per saber
   quina prova en concret s'està realitzant per ta d'actualitzar la
   matriu
344.             //del control de les accions fetes/no fetes
345.             accioseleccionada = accioo;
346.             targetseleccionat = targett;
347.             //a la següent linia activem el panell
348.             panelltext.SetActive(true);
349.         }
350.         //Aqueta funció s'encarrega de fer dues coses: Tancar
   el panell de text quan l'usuari prem el botó Tanca
351.         // i actualitzar la matriu del control de accions
352.         public void tancarpanelltext()
353.         {
354.             clasegenerica.GetInstance().ControlAccions[targetse
   leccionat - 1,accioseleccionada - 1] = 1;
355.             panelltext.SetActive(false);
356.         }
```

```
357.         //La funció activarpanelltextimatge(string textt,
   string imagepath, string respostacorrecta, int targett, int
   accioo)
358.         // s'encarrega d'executar la corutina que activarà el
   panell de text+imatge amb la informació adequada.
359.         public void activarpanelltextimatge(string textt, strin
   g imagepath, string respostacorrecta, int targett, int accioo)
360.         {
361.             //hi ha quatre variables d'entrada: La pregunta, la
   imatge i la resposta, el número de target i de prova.
362.             StartCoroutine(activarpanelltextimatgee(textt,
   imagepath, respostacorrecta, targett, accioo));
363.         }
364.         //La corutina activarpanelltextimatgee(string textt,
   string imagepath, string respostacorrecta, int targett, int
   accioo)
365.         //S'encarrega d'activar el panell amb la informació que
   rep com a entrada.
366.         public IEnumerator
   activarpanelltextimatgee(string textt, string imagepath, string
   respostacorrecta, int targett, int accioo)
367.         {
368.             accioseleccionada = accioo;
369.             targetseleccionat = targett;
370.             //primer introduim el text al panell.
371.             panelltextimatge.transform.GetChild(0).gameObject.G
   etComponent<Text>().text = textt;
372.             //seguidament descarreguem la imatge que s'ha de
   mostrar al panell i esperem la resposta.
373.             WWW
   www = new WWW("http://84.77.48.234:8523/miservidor/imagenes/" +
   imagepath);
374.             yield return www;
375.             //una vegada tenim la imatge, copiarem la textura a
   la imatge del panell.
376.             panelltextimatge.transform.GetChild(1).gameObject.G
   etComponent<RawImage>().texture = www.texture;
377.             panelltextimatge.transform.GetChild(2).gameObject.G
   etComponent<Text>().text = respostacorrecta;
378.             //finalment activem el panell
379.             panelltextimatge.SetActive(true);
380.         }
381.         //la funció tancarpanelltextimatge() tanca el panell de
   text+imatge quan l'alumne prem el botó i actualitza la matriu
   d'accions.
382.         public void tancarpanelltextimatge()
383.         {
384.             //a continuació comprovarem si la resposta que ha
   introduït l'alumne coincideix amb la correcta
385.             string resenviada=GameObject.Find("/canvasalumne/pa
   nellsaccions/panelltextimatge/InputField/Text").GetComponent<Tex
   t>().text;
386.             string rescorrecta= GameObject.Find("/canvasalumne/
   panellsaccions/panelltextimatge/respostacorrecta").GetComponent<
   Text>().text;
387.             if (resenviada==rescorrecta) //si coincideix
   s'asignarà un 2 a la matriu
388.             {
389.                 clasegenerica.GetInstance().ControlAccions[targ
   etseleccionat - 1, accioseleccionada - 1] = 2;
390.             }
```

```
391.             else // si no, un -2.
392.             {
393.                 clasegenerica.GetInstance().ControlAccions[targ
   etseleccionat - 1, accioseleccionada - 1] = -2;
394.             }
395.             //finalment esborrem el text que hagi introduit
   l'alumne i tanquem el panell
396.             GameObject.Find("/canvasalumne/panellsaccions/panel
   ltextimatge/InputField/Text").GetComponent<Text>().text = "
   ";
397.             panelltextimatge.SetActive(false);
398.
399.         }
400.         // la funció activarpanelltrivial(string pregunta,
   string a, string b, string c, string d,string correcta, int
   targett, int accioo)
401.         // activa el panell de trivial amb les variables que
   rep com a entrada
402.         public void activarpanelltrivial(string pregunta, strin
   g a, string b, string c, string d,string correcta, int targett,
   int accioo)
403.         {
404.             // les variables d'entrada son: la pregunta, les 4
   respostes possibles, la reposta correcta, i el número de target
   i provà cprresponents
405.             accioseleccionada = accioo;
406.             targetseleccionat = targett;
407.             panelltrivial.SetActive(true);
408.             panelltrivial.transform.GetChild(0).gameObject.GetC
   omponent<Text>().text = pregunta;
409.             panelltrivial.transform.GetChild(1).GetChild(0).gam
   eObject.GetComponent<Text>().text = a;
410.             panelltrivial.transform.GetChild(2).GetChild(0).gam
   eObject.GetComponent<Text>().text = b;
411.             panelltrivial.transform.GetChild(3).GetChild(0).gam
   eObject.GetComponent<Text>().text = c;
412.             panelltrivial.transform.GetChild(4).GetChild(0).gam
   eObject.GetComponent<Text>().text = d;
413.             inputresposta.Select();
414.             inputresposta.text = "";
415.             //A la següent linia assignem introduim la resposta
   correcta a la variable respostacorrectatrivial
416.             respostacorrectatrivial = correcta;
417.         }
418.         // A la funció marcarimatgee(string respostamarcada) el
   que fem es cambiar el color de la resposta marcada a vermell o
   verd
419.         // En funció de si l'alumne ha respós be la pregunta o
   no.
420.         public void marcarimatge(string respostamarcada)
421.         {
422.             //primer comprovem que no s'hagi marcat ja una
   altre resposta amb el valor del contadortrivial
423.
424.             if (contadortrivial==0)
425.             {
426.                 if (respostamarcada == respostacorrectatrivial)
427.                 {
428.                     panelltrivial.transform.Find("" + respostam
   arcada).GetComponent<UnityEngine.UI.Image>().color = Color.green
   ;
```

```
429.                     acertonotrivial = 1;
430.                 }
431.                 else
432.                 {
433.                     panelltrivial.transform.Find("" + respostam
    arcada).GetComponent<UnityEngine.UI.Image>().color = Color.red;
434.                     acertonotrivial = 0;
435.                 }
436.                 //A continuació incrementem el contador de
    manera que ja no permetim tornar a marcar una altra resposta
437.                 contadortrivial = 1;
438.             }
439.             else
440.             {
441.
442.             }
443.         }
444.         //A la funció tancarpanelltrivial() tancarem el panell
    de trivial quan l'alumne prem el botó tanca, i actualitza la
    matriu del
445.         // control de accions fetes/no fetes
446.         public void tancarpanelltrivial()
447.         {
448.             if (acertonotrivial==1) // si la resposta es
    correcta.
449.             {
450.                 clasegenerica.GetInstance().ControlAccions[targ
    etseleccionat - 1, accioseleccionada - 1] = 2;
451.                 //introduirem un 2 a la matriu.
452.             }
453.             else // si la resposta no es correcta
454.             {
455.                 clasegenerica.GetInstance().ControlAccions[targ
    etseleccionat - 1, accioseleccionada - 1] = -2;
456.                 //introduirem un -2 a la matriu.
457.             }
458.             // finalment tanquem el panell i tornem a posar
    tots els botons de resposta de color blanc.
459.             panelltrivial.SetActive(false);
460.             contadortrivial = 0;
461.             panelltrivial.transform.Find("a").GetComponent<Unit
    yEngine.UI.Image>().color = Color.white;
462.             panelltrivial.transform.Find("b").GetComponent<Unit
    yEngine.UI.Image>().color = Color.white;
463.             panelltrivial.transform.Find("c").GetComponent<Unit
    yEngine.UI.Image>().color = Color.white;
464.             panelltrivial.transform.Find("d").GetComponent<Unit
    yEngine.UI.Image>().color = Color.white;
465.         }
466.         //A la funció activarformulari(string linkformulari,
    int targett, int accioo) obrirem el link del formulari amb el
    navegador per defecte del
467.         // dispositiu utilitzat
468.         public void activarformulari(string linkformulari, int
    targett, int accioo)
469.         {
470.             //tal i com es pot veure, l'input està format pel
    link, el número de target i el número de acció
471.             accioseleccionada = accioo;
472.             targetseleccionat = targett;
473.             Application.OpenURL(linkformulari);
```

```
474.             //finalment actualitzem la matriu del control de
   accions introduint un 1 a la posició que portoca.
475.             clasegenerica.GetInstance().ControlAccions[targetse
   leccionat - 1, accioseleccionada - 1] = 1;
476.         }
477.         //la funció manarresultats s'encarrega de manar la
   puntuació de l'activitat. Primer es crea la conexió amb la base
   de dades i llavors manem
478.         // el nom de la escola, el número de grup, i la
   puntuació.
479.         public void manarresultats()
480.         {
481.             string escola = clasegenerica.GetInstance().nomesco
   la;
482.             int grup = clasegenerica.GetInstance().numerogrup;
483.             int sumatotal = clasegenerica.GetInstance().Control
   Accions.Cast<int>().Sum();
484.             Debug.Log(escola);
485.             Debug.Log(grup);
486.             Debug.Log(sumatotal);
487.             string url = "http://84.77.48.234:8523/miservidor/u
   serInsert.php";
488.             WWWForm formm = new WWWForm();
489.             formm.AddField("addescola", escola);
490.             formm.AddField("addgrup", grup);
491.             formm.AddField("addpuntuacio", sumatotal);
492.             WWW wwww= new WWW(url, formm);
493.         }
494.         //la funció setgridscores() s'encarrega de construir la
   graella de resultats cada vegada que accedim al panell de
   puntuacions
495.         public void setgridscores()
496.         {
497.             mostrarscor.SetActive(false);
498.             int[,] matriu = clasegenerica.GetInstance().Control
   Accions;
499.             gridd.SetActive(true);
500.             //a continuació comprovarem el número de proves de
   l'ctivitat. ja que depenent d'aquesta variable definirem una
   mida de les caselles o
501.             //una altra. Si hi ha més proves, les caselles
   seran més estretes, si hi hi ha més, més grans. La finalitat de
   tot això és aconseguir
502.             // que independentment del número de targets i
   accions de l'activitat, els resultats de la graella estiguin ben
   quadrats.
503.
504.             //      ____________________________________________
   _______      ____________________________________________________
505.             //      |         |         |         |         |
         |      |              |              |               |
506.             //      | EXEMPLE1| PROVA 1 | PROVA 2 | PROVA 3 |
   PROVA 4 |      |   EXEMPLE 2   |     PROVA 1    |     PROVA
   2    |
507.             //      |_________|_________|_________|_________|__
   _______|      |_______________|_______________|_______________
   |
508.             //      |         |         |         |         |
         |      |              |              |               |
```

```
509.             //      | TARGET
   1|    0    |    0    |    0    |    0    |      |    TARGET
   1   |        0        |       0        |
510.             //      |_________|_________|_________|_________|__
   _______|      |_________________|_________________|________________
   |
511.             //      |         |         |         |         |
         |      |               |                 |                |
512.             //      | TARGET
   2|    0    |    0    |    0    |    0    |      |    TARGET
   2   |        0        |       0        |
513.             //      |_________|_________|_________|_________|__
   _______|      |_________________|_________________|________________
   |
514.
515.
516.
517.             int tar = clasegenerica.GetInstance().naccions;
518.             if (tar==1)
519.             {
520.                 gridd.transform.GetChild(1).GetChild(0).GetChil
   d(0).gameObject.GetComponent<GridLayoutGroup>().cellSize = new V
   ector2(185,20);
521.
522.             }
523.             else if (tar==2)
524.             {
525.                 gridd.transform.GetChild(1).GetChild(0).GetChil
   d(0).gameObject.GetComponent<GridLayoutGroup>().cellSize = new V
   ector2(120, 20);
526.             }
527.             else if (tar == 3)
528.             {
529.                 gridd.transform.GetChild(1).GetChild(0).GetChil
   d(0).gameObject.GetComponent<GridLayoutGroup>().cellSize = new V
   ector2(90, 20);
530.             }
531.
532.             else if (tar == 4)
533.             {
534.                 gridd.transform.GetChild(1).GetChild(0).GetChil
   d(0).gameObject.GetComponent<GridLayoutGroup>().cellSize = new V
   ector2(70, 20);
535.             }
536.             else if (tar == 5)
537.             {
538.                 gridd.transform.GetChild(1).GetChild(0).GetChil
   d(0).gameObject.GetComponent<GridLayoutGroup>().cellSize = new V
   ector2(60, 20);
539.             }
540.             else
541.             {
542.                 gridd.transform.GetChild(1).GetChild(0).GetChil
   d(0).gameObject.GetComponent<GridLayoutGroup>().cellSize = new V
   ector2(50, 20);
543.             }
544.             // A continuació instanciarem les caselles de la
   graella
545.             GameObject
   objectiii = Instantiate(GameObject.Find("celldademostra"), new Ve
   ctor3(-1f, 1f, 0f), Quaternion.identity);
```

```
546.             objectiii.transform.name = "primeracelda";
547.             objectiii.transform.Find("texte").GetComponent<Text
   >().text = ("");
548.             objectiii.transform.parent = GameObject.Find("/canv
   asalumne/panellscore/scrvw/Viewport/Content").transform;
549.             int ss = 0;
550.             // les caselles s'omplen d'esquerra  dreta, així
   que comencem introduint els titols : "accio 1", "accio 2"..
   "accio n"
551.             while (ss<clasegenerica.GetInstance().naccions)
552.             {
553.                 int sss = ss + 1;
554.                 GameObject
   objectii = Instantiate(GameObject.Find("celldademostra"), new Vec
   tor3(-1f, 1f, 0f), Quaternion.identity);
555.                 objectii.transform.name = "Accio" + sss;
556.                 objectii.transform.Find("texte").GetComponent<T
   ext>().text = ("Accio " + sss);
557.                 objectii.transform.parent = GameObject.Find("/c
   anvasalumne/panellscore/scrvw/Viewport/Content").transform;
558.                 ss = ss + 1;
559.             }
560.             int ii = 0;
561.             // Amb el següent loop anirem recorrentla matriu
   del control de accions per files i columnes per tal de omplir la
   graella
562.             // dels resultats.
563.             while (ii< clasegenerica.GetInstance().ntarg)
564.             {
565.                 int iii = ii + 1;
566.                 int jj = 0;
567.                     int jjj = jj + 1;
568.                     GameObject
   objectiiii = Instantiate(GameObject.Find("celldademostra"), new V
   ector3(-1f, 1f, 0f), Quaternion.identity);
569.                     objectiiii.transform.name = "Target
   " + iii;
570.                     objectiiii.transform.Find("texte").GetCompo
   nent<Text>().text = ("Target " + iii);
571.                     objectiiii.transform.parent = GameObject.Fi
   nd("/canvasalumne/panellscore/scrvw/Viewport/Content").transform
   ;
572.                 while (jj< clasegenerica.GetInstance().naccions
   )
573.                 {
574.                     GameObject
   objectiiiii = Instantiate(GameObject.Find("celldademostra"), new
   Vector3(-1f, 1f, 0f), Quaternion.identity);
575.                     objectiiiii.transform.name = "" +ii+""+jj;
576.                     objectiiiii.transform.Find("texte").GetComp
   onent<Text>().text = (""+matriu[ii,jj]);
577.                     objectiiiii.transform.parent = GameObject.F
   ind("/canvasalumne/panellscore/scrvw/Viewport/Content").transfor
   m;
578.                     jj = jj + 1;
579.                 }
580.                 ii = ii + 1;
581.             }
582.
583.         }
```

```
584.         //la funció setgridscores() activa el panell d'a´vís
   amb el missatge que rep com a entrada. Aquesta funció crida a la
   corrutina
585.         //activarpanellaviss ja que només volem que el missatge
   es mostri duran 3 segons i després desaparegui automàticament.
586.         public void activarpanellavis (string misatge)
587.         {
588.             StartCoroutine(activarpanellaviss(misatge));
589.         }
590.         public IEnumerator activarpanellaviss(string misatge)
591.         {
592.             panellavisos.SetActive(true);
593.             panellavisos.transform.GetChild(0).GetComponent<Tex
   t>().text = misatge;
594.             yield return new WaitForSeconds(3);
595.             panellavisos.SetActive(false);
596.         }
597.     }
```

teachermanag.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using System;
6. using Vuforia;
7. using UnityEngine.SceneManagement;
8. using UnityEditor;
9. using UnityEngine;
10.      using Vuforia;
11.      using System.IO;
12.      using System.Linq;
13.      using System.Globalization;
14.      using UnityEngine.EventSystems;
15.
16.      public class teachermanag : MonoBehaviour //aquesta classe
   gestiona les accions dutes a terme pels administradors.
17.      {
18.          // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
19.          // ------------------------------------------------------
   VARIABLES---------------------------------------------------------
   ----------------------------
20.          // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
21.
22.          //primer declarem els dfiferents panells utilitzats per
   generar les partides
23.          public GameObject panelltext; // Panell constructor d'
   accions de text.
24.          public GameObject panelltextimatge; //Panell
   constructor d' accions de tex + imatge.
25.          public GameObject panellformulari; //Panell constructor
   d' accions formulari.
26.          public GameObject panelltrivial; //Panell constructor
   d' accions de trivial.
27.          public GameObject tipusaccions; // panell en el que es
   decideixen el tipus de proves de la gimcana.
```

```
28.         public GameObject panellfitxers; // panell per pujar
    fitxers al servidor directament.
29.         public GameObject panelldadesgen; //panell on
    s'introdueixen les dades generals de la gimcana.
30.         public GameObject escollirmode; // panell d'elecció
    entre generar una partida manualment o pujar fitxers al server
    directament.
31.         public GameObject panellfinal; // última finestra que
    s'bre quan ja s'ha generat l'activitat.
32.         public GameObject panellcoordenades;
33.
34.
35.         // ------------------------------------------------------
    ------------------------------------------------------------------
    ----------------------------
36.         // ------------------------------------------------------
    FUNCIONS-----------------------------------------------------------
    ----------------------------
37.         // ------------------------------------------------------
    ------------------------------------------------------------------
    ----------------------------
38.         // la funció introduir dadesgenerals() s'encarrega de
    generar un fitxer de text amb les dades generals de l'activitat.
39.         // Llavors truca a la corutina dadgenerals() que mana
    el fitxer al servidor.
40.         public void introduirgenerals()
41.         {
42.             int i = 0;
43.             //primer definirem una llista d'strings on
    guardarem tota la informació seleccionada als dropdowns del
    panell
44.             List<string> llistageneral = new List<string>();
45.             //acontinuació anem afegint aquesta informació a la
    llista.
46.             int s = GameObject.Find("/teachercanvas/dadesgen/pa
    neldrops/ntargg").GetComponent<Dropdown>().value + 1;
47.             int t=  GameObject.Find("/teachercanvas/dadesgen/pa
    neldrops/ncubss").GetComponent<Dropdown>().value + 1;
48.             llistageneral.Add(s.ToString());
49.             llistageneral.Add(t.ToString());
50.             llistageneral.Add((GameObject.Find("/teachercanvas/
    dadesgen/paneldrops/unaotot").GetComponent<Dropdown>().value).To
    String());
51.             llistageneral.Add((GameObject.Find("/teachercanvas/
    dadesgen/paneldrops/ordreono").GetComponent<Dropdown>().value).T
    oString());
52.             int contador = 0;
53.
54.             while (contador<s)
55.             {
56.                 llistageneral.Add(clasegenerica.GetInstance().m
    atriucoordenades[contador, 0].ToString("0.00000", System.Globali
    zation.CultureInfo.InvariantCulture) +"
    "+ clasegenerica.GetInstance().matriucoordenades[contador, 1].To
    String("0.00000", System.Globalization.CultureInfo.InvariantCult
    ure));
57.                 contador = contador + 1;
58.             }
59.             //a les següents quatre linies manarem les dades
    generals a la clase generica:
60.             //codi identificatiu de la gimcana
```

```
61.             clasegenerica.GetInstance().codicreador = GameObjec
   t.Find("/teachercanvas/dadesgen/paneldrops/inpcodi").GetComponen
   t<InputField>().text;
62.             //numero de targets
63.             clasegenerica.GetInstance().ntargcreador = s;
64.             // numero de proves per target
65.             clasegenerica.GetInstance().naccionscreador = t;
66.             // binari que defineix si s'ha de fer només una
   prova o totes
67.             clasegenerica.GetInstance().unaototescreador = Game
   Object.Find("/teachercanvas/dadesgen/paneldrops/unaotot").GetCom
   ponent<Dropdown>().value;
68.             // binari que defineix si els targets s'han de
   visitar en ordre
69.             clasegenerica.GetInstance().ordrecreador = GameObje
   ct.Find("/teachercanvas/dadesgen/paneldrops/ordreono").GetCompon
   ent<Dropdown>().value;
70.             // seguidament generem un fitxer de text on
   introduim les dades de la llista que hem creat previament
71.             StreamWriter
   writer = new StreamWriter(@"Assets/fitxersenviats/dadesgenerals"
   +@GameObject.Find("/teachercanvas/dadesgen/paneldrops/inpcodi").
   GetComponent<InputField>().text + @".txt");
72.             int j = 0;
73.             while (j < llistageneral.Count)
74.             {
75.                 writer.WriteLine(llistageneral[j]);
76.                 j = j + 1;
77.             }
78.             writer.Close();
79.             //finalment truquem a la corutina dadgenerals
80.             StartCoroutine(dadgenerals());
81.             tipusaccions.SetActive(true);
82.         }
83.         // la corutina dadgenerals() mana el fitxer de dades
   generals al servidor: recordem que el fitxer de dades generals
   rep el nom:
84.         // XXXXdadesgenerals.txt
85.         public IEnumerator dadgenerals()
86.         {
87.
88.             WWW
   fitxergeneral = new WWW("file:///Assets/fitxersenviats/dadesgene
   rals" + GameObject.Find("/teachercanvas/dadesgen/paneldrops/inpc
   odi").GetComponent<InputField>().text + ".txt");
89.             yield return fitxergeneral;
90.             //en les següents tres linies manarem el fitxer que
   acabem de copiar de la carpeta fitxers al servidor. Per fer-ho
   utilitzarem
91.             // la funció cargarachivotexto disponible a la
   carpeta principal del servidor
92.             WWWForm pujararxiu = new WWWForm();
93.             pujararxiu.AddBinaryData("file",
   fitxergeneral.bytes, "dadesgenerals"+ GameObject.Find("/teacherc
   anvas/dadesgen/paneldrops/inpcodi").GetComponent<InputField>().t
   ext+".txt", "text/plain");
94.             WWW
   upload = new WWW("http://84.77.48.234:8523/miservidor/cargararch
   ivotexto.php", pujararxiu);
95.             //finalment tancarem el panell de dadesgenerals
```

```
96.             GameObject.Find("/teachercanvas/dadesgen").SetActiv
   e(false);
97.             int j = 5;
98.             while (j >= clasegenerica.GetInstance().naccionscre
   ador)
99.             {
100.                GameObject.Find("/teachercanvas/tipusdeaccions/
   paneldrops/accio" + (j + 1).ToString()).SetActive(false);
101.                GameObject.Find("/teachercanvas/tipusdeaccions/
   paneldrops/textaccio" + (j + 1).ToString()).SetActive(false);
102.
103.                j = j - 1;
104.            }
105.        }
106.        //la funció carregar material obre un quadre de diàleg
   per seleccionar una foto que servirà com a textura pel material
   del cub
107.        // en el que estem treballant.
108.        public void carregarmaterial()
109.        {
110.            string pathh = EditorUtility.OpenFilePanel("Overwri
   te with jpg", "", "jpg");
111.            clasegenerica.GetInstance().materialaccio = pathh;
112.            //una vegada tenim l'adreça del nostre ordinador
   truquem a la corutina carregarmateriall()
113.
114.            StartCoroutine(carregarmateriall(pathh));
115.        }
116.        // la corrutina carregarmateriall s'encarrega de manar
   la imatge seleccionada en la funció carregarmaterial() al
   servidor ja que quan
117.        // l'alumne es vulgui baixar l'activitat totes les
   imatges les carregarà del servidor.
118.        public IEnumerator carregarmateriall(string pat)
119.        {
120.            WWW arxiumat = new WWW("file:///" + @pat);
121.            WWWForm postformimatgemat = new WWWForm();
122.            yield return postformimatgemat;
123.            postformimatgemat.AddBinaryData("file",
   arxiumat.bytes, Path.GetFileName(pat), "image/png");
124.            //a la següent linia es pot veure que per manar la
   imatge utilitzen la funció cargararchivo.pho del servidor.
125.            WWW
   upload = new WWW("http://84.77.48.234:8523/miservidor/cargararch
   ivo.php", postformimatgemat);
126.            //esperem fins que s'hag carregat
127.            yield return upload;
128.        }
129.        //la funció settipusaccions consulta al panell
   "tipusaccions" quin tipus de prova ha asignat l'administrador a
   cada cub. Recordem que
130.        // aquesta elecció es fa a través de dropdowns. LLavors
   consultem el valor que té el dropdown i l'afegim a la llista
   llistaccions de la
131.        // clasegenerica. D'aquesta manera obtenim una llista
   en la que emmagatzem quins tipus de accions ha seleccionar
   l'administrador
132.        public void settipusaccions()
133.        {
134.            int i = 0;
```

```
135.             while (i<clasegenerica.GetInstance().naccionscreado
   r)
136.             {
137.                 int ii = i + 1;
138.                 clasegenerica.GetInstance().llistaaccions.Add(G
   ameObject.Find("/teachercanvas/tipusdeaccions/paneldrops/accio"+
   ii.ToString()).GetComponent<Dropdown>().value);
139.                 i = i + 1;
140.             }
141.         }
142.         // La funció següentaccio(int indexx) serveix per
   tancar el panell de la prova que s'acaba de configurar i activa
   el següent. Per tal de saber
143.         // quins panells s'han d'activar aquesta funció
   consulta la llista tipusaccions de la clasegenerica
144.         public void seguentaccio(int indexx)
145.         {
146.             //la funció rep un index que es en realitat
   l'identificador del panell que s'ha de tancar i llavors
   desactiva el panell indicat.
147.             if (indexx==0) //desactivarem el panell
   tipusaccions
148.             {
149.                 tipusaccions.SetActive(false);
150.             }
151.             else if (indexx == 1) //desactivarem el panell
   constructor de proves de tìpus text
152.             {
153.                 panelltext.SetActive(false);
154.             }
155.             else if (indexx == 2) //desactivarem el panell
   constructor de proves de tìpus text+imatge
156.             {
157.                 panelltextimatge.SetActive(false);
158.             }
159.             else if (indexx == 3) //desactivarem el panell
   constructor de proves de tìpus formulari
160.             {
161.                 panellformulari.SetActive(false);
162.             }
163.             else //desactivarem el panell constructor de proves
   de tìpus trivial
164.             {
165.                 panelltrivial.SetActive(false);
166.             }
167.             //el funcionamente de la llista de accions funciona
   de la següent manera. Cada vegada que l' administrador acaba de
   programar una prova i prem el botó
168.             //continua, la acció que estaba fent s'elimina de
   la llista, de manera que el primer element passa a ser el que
   s'ha d'obrir a continuació, tal i com
169.             //es pot apreciar al següent condicional
170.             if (clasegenerica.GetInstance().llistaaccions.Count
   !=0)
171.             {
172.                 //si encara queden proves per programar
   entrarem al següent if
173.                if( clasegenerica.GetInstance().llistaaccions[0]
   ==0)
174.                 {
```

```
175.                    //si la següent acció que s'ha de programar
    es de tipus text, s'obrirà el corresponent panell constructor
    d'aquesta acció
176.                    clasegenerica.GetInstance().llistaaccions.R
    emoveAt(0);
177.                    panelltext.transform.Find("numaccio/Text").
    gameObject.GetComponent<Text>().text = ("Acció
    " + (clasegenerica.GetInstance().naccionscreador - clasegenerica
    .GetInstance().llistaaccions.Count).ToString());
178.                    InputField
    inputtext = panelltext.transform.Find("panelltext/InputField").G
    etComponent<InputField>();
179.                    inputtext.Select();
180.                    inputtext.text = "";
181.                    panelltext.SetActive(true);
182.                }
183.                else if (clasegenerica.GetInstance().llistaacci
    ons[0] == 1)
184.                {
185.                    //si la següent acció que s'ha de programar
    es de tipus text+imatge, s'obrirà el corresponent panell
    constructor d'aquesta acció
186.                    //però abans haurem de preparar la scroll
    view que ens permetrà carregar les imatges i introduir les
    respostes
187.                    clasegenerica.GetInstance().llistaaccions.R
    emoveAt(0);
188.                    panelltextimatge.transform.Find("numaccio/T
    ext").gameObject.GetComponent<Text>().text = ("Acció
    " + (clasegenerica.GetInstance().naccionscreador - clasegenerica
    .GetInstance().llistaaccions.Count).ToString());
189.                    int g = 0;
190.                    panelltextimatge.SetActive(true);
191.                    //en les següents linies s'instancien les
    "caselles" que ompliran el contenidor de la scroll view
192.                    //Primer afegim els encapçalats "accio"
    "imatge" "estat" i "resposta"
193.                    GameObject
    itemacciotit = Instantiate(GameObject.Find("/plantilles/plantill
    aaccio"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
194.                    itemacciotit.transform.name = "accio";
195.                    itemacciotit.transform.Find("Text").GetComp
    onent<Text>().text = ("Acció");
196.                    itemacciotit.GetComponent<UnityEngine.UI.Im
    age>().color= new Color(1f, 0.9233f, 0f, 1f);
197.                    itemacciotit.transform.parent = GameObject.
    Find("/teachercanvas/constructors/constructorimatgetext/panellte
    xtimatge/ScrollView/Viewport/Content").transform;
198.                    GameObject
    itemcarregartit = Instantiate(GameObject.Find("/plantilles/plant
    illaaccio"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
199.                    itemcarregartit.transform.name = "Imatge" ;
200.                    itemcarregartit.transform.Find("Text").GetC
    omponent<Text>().text = ("Imatge");
201.                    itemcarregartit.GetComponent<UnityEngine.UI
    .Image>().color = new Color(1f, 0.9233f, 0f, 1f);
202.                    itemcarregartit.transform.parent = GameObje
    ct.Find("/teachercanvas/constructors/constructorimatgetext/panel
    ltextimatge/ScrollView/Viewport/Content").transform;
```

```
203.                     GameObject
    itemcarregattit = Instantiate(GameObject.Find("/plantilles/plant
    illaaccio"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
204.                     itemcarregattit.transform.name = "Estat" ;
205.                     itemcarregattit.transform.Find("Text").GetC
    omponent<Text>().text = ("Estat");
206.                     itemcarregattit.GetComponent<UnityEngine.UI
    .Image>().color = new Color(1f, 0.9233f, 0f, 1f);
207.                     itemcarregattit.transform.parent = GameObje
    ct.Find("/teachercanvas/constructors/constructorimatgetext/panel
    ltextimatge/ScrollView/Viewport/Content").transform;
208.                     GameObject
    itemrespostatit = Instantiate(GameObject.Find("/plantilles/plant
    illaaccio"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
209.                     itemrespostatit.transform.name = "Resposta"
     ;
210.                     itemrespostatit.transform.Find("Text").GetC
    omponent<Text>().text = ("Resposta");
211.                     itemrespostatit.GetComponent<UnityEngine.UI
    .Image>().color = new Color(1f, 0.9233f, 0f, 1f);
212.                     itemrespostatit.transform.parent = GameObje
    ct.Find("/teachercanvas/constructors/constructorimatgetext/panel
    ltextimatge/ScrollView/Viewport/Content").transform;
213.                     //el següent loop emplenarà tantes files de
    caselles com targets hi hagi a l'activitat. EL número de targets
    es consulta de la clasegenerica.
214.                     while (g<clasegenerica.GetInstance().ntargc
    reador)
215.                     {
216.                         int gg = g + 1;
217.                         //casella indicant el número de acció.
218.                         GameObject
    itemaccio = Instantiate(GameObject.Find("/plantilles/plantillaac
    cio"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
219.                         itemaccio.transform.name = "accio" + gg
    .ToString();
220.                         itemaccio.transform.Find("Text").GetCom
    ponent<Text>().text = ("Acció "+gg.ToString());
221.                         itemaccio.transform.parent = GameObject
    .Find("/teachercanvas/constructors/constructorimatgetext/panellt
    extimatge/ScrollView/Viewport/Content").transform;
222.                         //casella per carregar imatges (té un
    component de botó)
223.                         GameObject
    itemcarregar = Instantiate(GameObject.Find("/plantilles/plantill
    acarregar"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
224.                         itemcarregar.transform.name = "carregar
    " + gg.ToString();
225.                         itemcarregar.transform.parent = GameObj
    ect.Find("/teachercanvas/constructors/constructorimatgetext/pane
    lltextimatge/ScrollView/Viewport/Content").transform;
226.                         //casella indicant l'estat de càrrega
    de la imatge
227.                         GameObject
    itempendent = Instantiate(GameObject.Find("/plantilles/plantilla
    check"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
228.                         itempendent.transform.name = "pendent"
    + gg.ToString();
229.                         itempendent.transform.parent = GameObje
    ct.Find("/teachercanvas/constructors/constructorimatgetext/panel
    ltextimatge/ScrollView/Viewport/Content").transform;
```

```
230.                        //casella per introduir la resposta (és
   un input field)
231.                        GameObject
   itemresposta = Instantiate(GameObject.Find("/plantilles/plantill
   aresposta"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
232.                        itemresposta.transform.name = "resposta
   " + gg.ToString();
233.                        itemresposta.transform.parent = GameObj
   ect.Find("/teachercanvas/constructors/constructorimatgetext/pane
   lltextimatge/ScrollView/Viewport/Content").transform;
234.                        g = g + 1;
235.                    }
236.                    //esborrem també el camp d'entrada per la
   pregunta
237.                    InputField
   inputtextimatge = panelltextimatge.transform.Find("panelltextima
   tge/InputField").GetComponent<InputField>();
238.                    inputtextimatge.Select();
239.                    inputtextimatge.text = "";
240.                }
241.                else if (clasegenerica.GetInstance().llistaacci
   ons[0] == 2)
242.                {
243.                    //si la següent acció que s'ha de programar
   es de tipus formulari, s'obrirà el corresponent panell
   constructor d'aquesta acció
244.                    //al igual que en els casos anteriors,
   resetejarem el camp d'entrada ja que si prèviament s'ha
   configurat una acció de formulari
245.                    //aquest camp romandrà encara omplert
246.                    clasegenerica.GetInstance().llistaaccions.R
   emoveAt(0);
247.                    panellformulari.transform.Find("numaccio/Te
   xt").gameObject.GetComponent<Text>().text = ("Acció
   " + (clasegenerica.GetInstance().naccionscreador - clasegenerica
   .GetInstance().llistaaccions.Count).ToString());
248.                    InputField
   inputform = panellformulari.transform.Find("panellformulari/Inpu
   tField").GetComponent<InputField>();
249.                    inputform.Select();
250.                    inputform.text = "";
251.                    panellformulari.SetActive(true);

252.                }
253.                else if (clasegenerica.GetInstance().llistaacci
   ons[0] == 3)
254.                {
255.                    //si la següent acció que s'ha de programar
   es de tipus text+imatge, s'obrirà el corresponent panell
   constructor d'aquesta acció
256.                    //però abans haurem de preparar la scroll
   view que ens permetrà introduir les preguntes i respostes
257.                    clasegenerica.GetInstance().llistaaccions.R
   emoveAt(0);
258.                    panelltrivial.transform.Find("numaccio/Text
   ").gameObject.GetComponent<Text>().text = ("Acció
   " + (clasegenerica.GetInstance().naccionscreador - clasegenerica
   .GetInstance().llistaaccions.Count).ToString());
259.                    int d = 0;
260.                    while (d < clasegenerica.GetInstance().ntar
   gcreador)
```

```
261.                     {
262.                         int dd = d + 1;
263.                         GameObject
    itempregunta = Instantiate(GameObject.Find("/plantilles/plantill
    aaccio"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
264.                         itempregunta.transform.name = "pregunt"
     + dd.ToString();
265.                         itempregunta.transform.Find("Text").Get
    Component<Text>().text = ("Pregunta " + dd.ToString());
266.                         itempregunta.transform.parent = GameObj
    ect.Find("/teachercanvas/constructors/constructortrivial/panellt
    rivial/ScrollView/Viewport/Content").transform;
267.                         itempregunta.GetComponent<UnityEngine.U
    I.Image>().color = new Color(1f, 0.9233f, 0f, 1f);
268.
269.                         GameObject
    itempreg = Instantiate(GameObject.Find("/plantilles/plantillares
    posta"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
270.                         itempreg.transform.name = "pregunta" +
    dd.ToString();
271.                         itempreg.GetComponent<InputField>().Sel
    ect();
272.                         itempreg.GetComponent<InputField>().tex
    t = "Pregunta";
273.                         itempreg.transform.parent = GameObject.
    Find("/teachercanvas/constructors/constructortrivial/panelltrivi
    al/ScrollView/Viewport/Content").transform;
274.
275.                         GameObject
    itemrespostaa = Instantiate(GameObject.Find("/plantilles/plantil
    laresposta"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
276.                         itemrespostaa.transform.name = "respost
    aa" + dd.ToString();
277.                         itemrespostaa.GetComponent<InputField>(
    ).Select();
278.                         itemrespostaa.GetComponent<InputField>(
    ).text = "Resposta a";
279.                         itemrespostaa.transform.parent = GameOb
    ject.Find("/teachercanvas/constructors/constructortrivial/panell
    trivial/ScrollView/Viewport/Content").transform;
280.
281.                         GameObject
    itemrespostab = Instantiate(GameObject.Find("/plantilles/plantil
    laresposta"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
282.                         itemrespostab.transform.name = "respost
    ab" + dd.ToString();
283.                         itemrespostab.GetComponent<InputField>(
    ).Select();
284.                         itemrespostab.GetComponent<InputField>(
    ).text = "Resposta b";
285.                         itemrespostab.transform.parent = GameOb
    ject.Find("/teachercanvas/constructors/constructortrivial/panell
    trivial/ScrollView/Viewport/Content").transform;
286.
287.                         GameObject
    itemrespostac = Instantiate(GameObject.Find("/plantilles/plantil
    laresposta"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
288.                         itemrespostac.transform.name = "respost
    ac" + dd.ToString();
289.                         itemrespostac.GetComponent<InputField>(
    ).Select();
```

```
290.                         itemrespostac.GetComponent<InputField>(
    ).text = "Resposta c";
291.                         itemrespostac.transform.parent = GameOb
    ject.Find("/teachercanvas/constructors/constructortrivial/panell
    trivial/ScrollView/Viewport/Content").transform;
292.
293.                         GameObject
    itemrespostad = Instantiate(GameObject.Find("/plantilles/plantil
    laresposta"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
294.                         itemrespostad.transform.name = "respost
    ad" + dd.ToString();
295.                         itemrespostad.GetComponent<InputField>(
    ).Select();
296.                         itemrespostad.GetComponent<InputField>(
    ).text = "Resposta d";
297.                         itemrespostad.transform.parent = GameOb
    ject.Find("/teachercanvas/constructors/constructortrivial/panell
    trivial/ScrollView/Viewport/Content").transform;
298.
299.                         GameObject
    itemrespostacorr = Instantiate(GameObject.Find("/plantilles/plan
    tillaopcions"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
300.                         itemrespostacorr.transform.name = "resp
    ostacorrecta" + dd.ToString();
301.                         itemrespostacorr.transform.parent = Gam
    eObject.Find("/teachercanvas/constructors/constructortrivial/pan
    elltrivial/ScrollView/Viewport/Content").transform;
302.                         d = d + 1;
303.                     }
304.                     panelltrivial.SetActive(true);

305.                 }
306.             }
307.             else
308.             {
309.                 //si ja no queden més proves, s'activarà el
    panell final que permetrà retornar al menú principal del joc.
310.                 panellfinal.SetActive(true);
311.                 panellfinal.transform.Find("Panel/codi").GetCom
    ponent<Text>().text = "CODI:
    "+clasegenerica.GetInstance().codicreador;
312.             }
313.         }
314.         // la funció loadmainscena simplement carrega la escena
    principal del joc, on es pot escollir entre el perfil
    d'administrador i alumne
315.         public void loadmainscene()
316.         {
317.             SceneManager.LoadScene("SampleScene");
318.             panellfinal.SetActive(false);
319.         }
320.         // la funció crearacciotext, com el seu nom indica s'
    engarrega de generar un fitxer en local amb les dades de la
    prova, i després
321.         // truca a la corutina manaraccio per tal d'enviar el
    fitxer al servidor
322.         public void crearacciotext()
323.         {
324.             //primer gnerem una llista on anirem afegint les
    dades
325.             List<string> dadestext= new List<string>();
```

```
326.             dadestext.Add("TEXT"); //capçalera del fitxer
327.             dadestext.Add(Path.GetFileName(clasegenerica.GetIns
   tance().materialaccio)); // material
328.             dadestext.Add(panelltext.transform.Find("panelltext
   /InputField/Text").gameObject.GetComponent<Text>().text); //text
   de l'acció
329.             //a continuació generem el fitxer
330.             StreamWriter
   arxiudadestext = new StreamWriter(@"Assets/fitxersenviats/"+ (cl
   asegenerica.GetInstance().naccionscreador - clasegenerica.GetIns
   tance().llistaaccions.Count).ToString() + clasegenerica.GetInsta
   nce().codicreador+".txt");
331.             int jj = 0;
332.             //i amb el següent loop l'anem omplint
333.             while (jj < dadestext.Count)
334.             {
335.                 arxiudadestext.WriteLine(dadestext[jj]);
336.                 jj = jj + 1;
337.             }
338.             arxiudadestext.Close();
339.             // es truca la corutina manaraccio()
340.             StartCoroutine(manaraccio("Assets/fitxersenviats/"
   +
341.                 "" + (clasegenerica.GetInstance().naccionscread
   or - clasegenerica.GetInstance().llistaaccions.Count).ToString()
    + clasegenerica.GetInstance().codicreador + ".txt"));
342.         }
343.         // la funció crearacciotext, com el seu nom indica s'
   engarrega de generar un fitxer en local amb les dades de la
   prova, i després
344.         // truca a la corutina manaraccio per tal d'enviar el
   fitxer al servidor
345.         public void crearaccioformulari()
346.         {
347.             //primer gnerem una llista on anirem afegint les
   dades
348.             List<string> dadesformulari = new List<string>();
349.             dadesformulari.Add("FORMULARI"); //capçalera del
   fitxer
350.             dadesformulari.Add(Path.GetFileName(clasegenerica.G
   etInstance().materialaccio)); //material
351.             dadesformulari.Add(panellformulari.transform.Find("
   panellformulari/InputField/Text").gameObject.GetComponent<Text>(
   ).text); //link del formulari
352.             //a continuació generem el fitxer
353.             StreamWriter
   arxiudadesformulari = new StreamWriter(@"Assets/fitxersenviats/"
    + (clasegenerica.GetInstance().naccionscreador - clasegenerica.
   GetInstance().llistaaccions.Count).ToString() + clasegenerica.Ge
   tInstance().codicreador + ".txt");
354.             int jj = 0;
355.             //i amb el següent loop l'anem omplint
356.             while (jj < dadesformulari.Count)
357.             {
358.                 arxiudadesformulari.WriteLine(dadesformulari[jj
   ]);
359.                 jj = jj + 1;
360.             }
361.             arxiudadesformulari.Close();
362.             // es truca la corutina manaraccio()
```

```
363.             StartCoroutine(manaraccio("Assets/fitxersenviats/"
   + (clasegenerica.GetInstance().naccionscreador - clasegenerica.G
   etInstance().llistaaccions.Count).ToString() + clasegenerica.Get
   Instance().codicreador + ".txt"));
364.         }
365.         // la funció crearacciotext, com el seu nom indica s'
   engarrega de generar un fitxer en local amb les dades de la
   prova, i després
366.         // truca a la corutina manaraccio per tal d'enviar el
   fitxer al servidor
367.         public void crearacciotrivial()
368.         {
369.             //primer gnerem una llista on anirem afegint les
   dades
370.             List<string> dadestrivial= new List<string>();
371.             dadestrivial.Add("TRIVIAL"); //capçalera del fitxer
372.             dadestrivial.Add(Path.GetFileName(clasegenerica.Get
   Instance().materialaccio)); //material
373.             int i = 0;
374.             // seguidament anem afegint les preguntes,
   respostes i resposta correcta de cada pregunta, de forma
   ordenada
375.             while (i<clasegenerica.GetInstance().ntargcreador)
376.             {
377.                 int ii = i + 1;
378.                 dadestrivial.Add(GameObject.Find("/teachercanva
   s/constructors/constructortrivial/panelltrivial/ScrollView/Viewp
   ort/Content/pregunta"+ii).GetComponent<InputField>().text);
379.                 dadestrivial.Add(GameObject.Find("/teachercanva
   s/constructors/constructortrivial/panelltrivial/ScrollView/Viewp
   ort/Content/respostaa" + ii).GetComponent<InputField>().text);
380.                 dadestrivial.Add(GameObject.Find("/teachercanva
   s/constructors/constructortrivial/panelltrivial/ScrollView/Viewp
   ort/Content/respostab" + ii).GetComponent<InputField>().text);
381.                 dadestrivial.Add(GameObject.Find("/teachercanva
   s/constructors/constructortrivial/panelltrivial/ScrollView/Viewp
   ort/Content/respostac" + ii).GetComponent<InputField>().text);
382.                 dadestrivial.Add(GameObject.Find("/teachercanva
   s/constructors/constructortrivial/panelltrivial/ScrollView/Viewp
   ort/Content/respostad" + ii).GetComponent<InputField>().text);
383.                 Dropdown
   triv = GameObject.Find("/teachercanvas/constructors/constructort
   rivial/panelltrivial/ScrollView/Viewport/Content/respostacorrect
   a" + ii).GetComponent<Dropdown>();
384.                 string ultimalletra = triv.options[triv.GetComp
   onent<Dropdown>().value].text;
385.                 dadestrivial.Add(ultimalletra.Substring(ultimal
   letra.Length - 1, 1));
386.                 i = i + 1;
387.             }
388.             //seguidament generem el fitxer de text
389.             StreamWriter
   arxiudadestrivial = new StreamWriter(@"Assets/fitxersenviats/" +
    (clasegenerica.GetInstance().naccionscreador - clasegenerica.Ge
   tInstance().llistaaccions.Count).ToString() + clasegenerica.GetI
   nstance().codicreador + ".txt");
390.             int jjj = 0;
391.             //i anem afegint les dades de la llista
392.             while (jjj < dadestrivial.Count)
393.             {
394.                 arxiudadestrivial.WriteLine(dadestrivial[jjj]);
```

```
395.                 jjj = jjj + 1;
396.             }
397.             arxiudadestrivial.Close();
398.             StartCoroutine(manaraccio("Assets/fitxersenviats/"
   + (clasegenerica.GetInstance().naccionscreador - clasegenerica.G
   etInstance().llistaaccions.Count).ToString() + clasegenerica.Get
   Instance().codicreador + ".txt"));
399.             foreach (Transform
   child in GameObject.Find("teachercanvas/constructors/constructor
   trivial/panelltrivial/ScrollView/Viewport/Content").transform)
400.             {
401.                 Destroy(child.gameObject);
402.             }
403.
404.         }
405.         //la funció crearnoutipusdeprova() serviría per generar
   el fitxer de la prova i manar-lo al servidor
406.         public void crearnoutipusdeprova()
407.         {
408.             //Primer definim una llista que contindrà tota la
   informació per després emplenar el fitxer
409.             List<string> dadesnovaaccio = new List<string>();
410.             dadesnovaaccio.Add("NOU TIPUS DE
   ACCIÓ"); //capçalera del fitxer
411.             dadesnovaaccio.Add(Path.GetFileName(clasegenerica.G
   etInstance().materialaccio)); //material
412.             int i = 0;
413.             // seguidament anem afegint la informació a la
   llista
414.             while (i < clasegenerica.GetInstance().ntargcreador
   )
415.             {
416.                 dadesnovaaccio.Add("ANAR AFEGINT LA INFORMACIÓ
   DE LA PROVA");
417.                 i = i + 1;
418.             }
419.             //seguidament generem el fitxer de text
420.             StreamWriter
   arxiudadesnovaprova = new StreamWriter(@"Assets/fitxersenviats/"
    + (clasegenerica.GetInstance().naccionscreador - clasegenerica.
   GetInstance().llistaaccions.Count).ToString() + clasegenerica.Ge
   tInstance().codicreador + ".txt");
421.             int jjj = 0;
422.             //i anem afegint les dades de la llista
423.             while (jjj < dadesnovaaccio.Count)
424.             {
425.                 arxiudadesnovaprova.WriteLine(dadesnovaaccio[jj
   j]);
426.                 jjj = jjj + 1;
427.             }
428.             arxiudadesnovaprova.Close();
429.             StartCoroutine(manaraccio("Assets/fitxersenviats/"
   + (clasegenerica.GetInstance().naccionscreador - clasegenerica.G
   etInstance().llistaaccions.Count).ToString() + clasegenerica.Get
   Instance().codicreador + ".txt"));
430.         }
431.         //la funció detectarboto() serveix per identificar a
   quin target pertany el botó que s'ha polsat per carregar la
   imatge al contructor
432.         //d'imatge+text
433.         public void detectarboto()
```

```
434.         {
435.             //primer identifiquem el nom de l'objecte pulsat
436.             string lastt = EventSystem.current.currentSelectedG
   ameObject.name;
437.             string lastWord = lastt.Substring(lastt.Length - 1,
    1);
438.             Debug.Log(lastWord);
439.             int numberr = Int32.Parse(lastWord);
440.             int numberrr = numberr - 1;
441.             //i després cridem a la corutina detectarbotoo(nom)
442.             StartCoroutine(detectarbotoo(numberrr));
443.         }
444.         //la corutina detectarbotoo obrirà un quadre de diàleg
   per carregar una imatge cada vegada que premem sobre un botó de
   càrrega de imatges
445.         // del panell. Seguidament envia les imatges al servido
   i mana el nom de la imatge a la llista llistaimatrges de la
   clasegenerica.
446.         public IEnumerator detectarbotoo(int num)
447.         {
448.             int numm = num + 1;
449.             //obrim el quadre de diàleg i carreguem la imatge
450.             string pathhimatge = EditorUtility.OpenFilePanel("O
   verwrite with jpg", "", "jpg");
451.             WWW
   arxiuimatg = new WWW("file:///" + @pathhimatge);
452.             WWWForm postformimatge = new WWWForm();
453.             yield return postformimatge;
454.             postformimatge.AddBinaryData("file",
   arxiuimatg.bytes, Path.GetFileName(pathhimatge), "image/png");
455.             string nomarxiu = Path.GetFileName(pathhimatge);
456.             //guardem a la llista el nom de la imatge
457.             clasegenerica.GetInstance().llistaimatges[num] = no
   marxiu;
458.             Debug.Log(clasegenerica.GetInstance().llistaimatges
   [num]);
459.             //manem la imatge al servidor
460.             WWW
   upload = new WWW("http://84.77.48.234:8523/miservidor/cargararch
   ivo.php", postformimatge);
461.             yield return upload;
462.             Debug.Log(clasegenerica.GetInstance().llistaimatges
   [16]);
463.             yield return null;
464.             //actualitzem l'estat
465.             panelltextimatge.transform.Find("panelltextimatge/S
   crollView/Viewport/Content/pendent" + numm.ToString() + "/penden
   t").gameObject.SetActive(false);
466.             panelltextimatge.transform.Find("panelltextimatge/S
   crollView/Viewport/Content/pendent" + numm.ToString() + "/carreg
   at").gameObject.SetActive(true);
467.
468.         }
469.         // la funció crearacciotext, com el seu nom indica s'
   engarrega de generar un fitxer en local amb les dades de la
   prova, i després
470.         // truca a la corutina manaraccio per tal d'enviar el
   fitxer al servidor
471.         public void crearacciotextimatge()
472.         {
473.             //comencem creant la llista
```

```
474.             List<string> dadestextimatge = new List<string>();
475.             dadestextimatge.Add("TEXT+IMATGE"); //capçalera
476.             dadestextimatge.Add(Path.GetFileName(clasegenerica.
   GetInstance().materialaccio)); //material
477.             dadestextimatge.Add(panelltextimatge.transform.Find
   ("panelltextimatge/InputField/Text").gameObject.GetComponent<Tex
   t>().text); //pregunta
478.             string[] listaimatgess = clasegenerica.GetInstance(
   ).llistaimatges;
479.             int i = 0;
480.             //després anirem carregant la llista de imatges a
   la llista que acabem de crear
481.             while (i< clasegenerica.GetInstance().ntargcreador)
482.             {
483.                 dadestextimatge.Add(listaimatgess[i]);
484.                 i = i + 1;
485.             }
486.             int j = 0;
487.             //per acabar afegim les respostes introduides al
   camp d'entrada del panell
488.             while (j < clasegenerica.GetInstance().ntargcreador
   )
489.             {
490.                 int jj = j + 1;
491.                 dadestextimatge.Add(GameObject.Find("teachercan
   vas/constructors/constructorimatgetext/panelltextimatge/ScrollVi
   ew/Viewport/Content/resposta" + jj.ToString()+"/Text").GetCompon
   ent<Text>().text);
492.                 j = j + 1;
493.             }
494.             //generem un fitxer en local
495.             StreamWriter
   arxiudadestextimatge = new StreamWriter(@"Assets/fitxersenviats/
   " + (clasegenerica.GetInstance().naccionscreador - clasegenerica
   .GetInstance().llistaaccions.Count).ToString() + clasegenerica.G
   etInstance().codicreador + ".txt");
496.             int jjj = 0;
497.             while (jjj < dadestextimatge.Count)
498.             {
499.                 arxiudadestextimatge.WriteLine(dadestextimatge[
   jjj]);
500.                 jjj = jjj + 1;
501.             }
502.             arxiudadestextimatge.Close();
503.             //cridem a la corutina manaraccio per manar el
   fitxer al servidor
504.
505.             StartCoroutine(manaraccio("Assets/fitxersenviats/"
   + (clasegenerica.GetInstance().naccionscreador - clasegenerica.G
   etInstance().llistaaccions.Count).ToString() + clasegenerica.Get
   Instance().codicreador + ".txt"));
506.             foreach (Transform
   child in GameObject.Find("teachercanvas/constructors/constructor
   imatgetext/panelltextimatge/ScrollView/Viewport/Content").transf
   orm)
507.             {
508.                 Destroy(child.gameObject);
509.             }
510.         }
```

```
511.         //la corrutina manaraccio s'encarrega de manar el ftxer
   al servidor que rep en el camp d'entrada com una string. Aquest
   fitxer es troba
512.         //a la carpeta "fitxers" en local, així que només s'ha
   de agafar aquest fitxer i pujar-lo.
513.         public IEnumerator manaraccio(string nomarxiu)
514.         {
515.             WWW fitxeraccio = new WWW("file:///"+@nomarxiu);
516.             yield return fitxeraccio;
517.             WWWForm pujararxiuaccio = new WWWForm();
518.             yield return pujararxiuaccio;
519.             pujararxiuaccio.AddBinaryData("file",
   fitxeraccio.bytes, Path.GetFileName(nomarxiu), "text/plain");
520.             WWW
   uploadaccio = new WWW("http://84.77.48.234:8523/miservidor/carga
   rarchivotexto.php", pujararxiuaccio);
521.         }
522.         //Al métode start es generen dos array de 20 elements
   cadadcun (número màxim de targets) per guardar les imatges  i
   respostas quan estem
523.         //generant una prova del tipus text+imatge. Inicialment
   s'emplenen amb zeros
524.         void Start()
525.         {
526.             clasegenerica.GetInstance().llistaimatges = new str
   ing[20];
527.             clasegenerica.GetInstance().llistarespostes = new s
   tring[20];
528.             int i = 0;
529.             while (i <= 19)
530.             {
531.                 clasegenerica.GetInstance().llistaimatges[i] =
   "0";
532.                 clasegenerica.GetInstance().llistarespostes[i]
   = "0";
533.                 i = i + 1;
534.             }
535.         }
536.         //la funció loadfile() obre un quadre de diàleg per
   seleccionar un fitxer que vulguem manar al servidor. S'utilitza
   quan l'administrador
537.         //ja coneix com funciona el sistema de fitxers de
   l'aplicació i decideix manar els fitxers directament.
538.         public void loadafile()
539.         {
540.             string pathhh = EditorUtility.OpenFilePanel("Overwr
   ite with txt", "", "txt");
541.             StartCoroutine(manaraccio(pathhh));
542.             GameObject
   itemfile = Instantiate(GameObject.Find("/plantilles/plantillaacc
   io"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
543.             itemfile.transform.name = Path.GetFileName(pathhh);
544.             itemfile.transform.Find("Text").GetComponent<Text>(
   ).text = (Path.GetFileName(pathhh));
545.             itemfile.transform.parent = GameObject.Find("/teach
   ercanvas/carregarfitxers/panellcarrega/ScrollView/Viewport/Conte
   nt").transform;
546.         }
547.         //la funció loadimage() obre un quadre de diàleg per
   seleccionar una imatge que vulguem manar al servidor. S'utilitza
   quan l'administrador
```

```
548.         //ja coneix com funciona el sistema de fitxers de
   l'aplicació i decideix manar els fitxers directament.
549.         public void loadaimage()
550.         {
551.             string pathhhh = EditorUtility.OpenFilePanel("Overw
   rite with jpg", "", "jpg");
552.             StartCoroutine(carregarmateriall(pathhhh));
553.             GameObject
   itemfile = Instantiate(GameObject.Find("/plantilles/plantillaacc
   io"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
554.             itemfile.transform.name = Path.GetFileName(pathhhh)
   ;
555.             itemfile.transform.Find("Text").GetComponent<Text>(
   ).text = (Path.GetFileName(pathhhh));
556.             itemfile.transform.parent = GameObject.Find("/teach
   ercanvas/carregarfitxers/panellcarrega/ScrollVieww/Viewport/Cont
   ent").transform;
557.         }
558.         //la funció activar panell coordenades acriva el panell
   de coordenades per tal de poder introduir la latitud i longitud
   dels targets
559.         public void activarpanellcoordenades()
560.         {
561.             panellcoordenades.SetActive(true);
562.             int i = 0;
563.             while (i<(GameObject.Find("teachercanvas/dadesgen/p
   aneldrops/ntargg").GetComponent<Dropdown>().value+1))
564.             {
565.                 int ii = i + 1;
566.                 //Primer afegim els encapçalats "accio"
   "imatge" "estat" i "resposta"
567.                 GameObject
   numerotarget = Instantiate(GameObject.Find("/plantilles/plantill
   aaccio"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
568.                 numerotarget.transform.name = "accio"+ii.ToStri
   ng();
569.                 numerotarget.transform.Find("Text").GetComponen
   t<Text>().text = ("Acció "+ii.ToString());
570.                 numerotarget.GetComponent<UnityEngine.UI.Image>
   ().color = new Color(1f, 0.9233f, 0f, 1f);
571.                 numerotarget.transform.parent = GameObject.Find
   ("/teachercanvas/dadesgen/coordenades/panellcoordenades/ScrollVi
   ew/Viewport/Content").transform;
572.                 GameObject
   latitude = Instantiate(GameObject.Find("/plantilles/plantillares
   posta"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
573.                 latitude.transform.name = "latitude"+ii.ToStrin
   g();
574.                 InputField
   inputlat = latitude.GetComponent<InputField>();
575.                 inputlat.Select();
576.                 inputlat.text = "latitud";
577.                 latitude.transform.parent = GameObject.Find("/t
   eachercanvas/dadesgen/coordenades/panellcoordenades/ScrollView/V
   iewport/Content").transform;
578.                 GameObject
   longitude = Instantiate(GameObject.Find("/plantilles/plantillare
   sposta"), new Vector3(-1f, 1f, 0f), Quaternion.identity);
579.                 longitude.transform.name = "longitude" + ii.ToS
   tring();
```

```
580.                 InputField
    inputlon = longitude.GetComponent<InputField>();
581.                 inputlon.Select();
582.                 inputlon.text = "longitud";
583.                 longitude.transform.parent = GameObject.Find("/
    teachercanvas/dadesgen/coordenades/panellcoordenades/ScrollView/
    Viewport/Content").transform;
584.                 i = i + 1;
585.             }
586.         }
587.         //la funcio tancarpanellcoordenades introduiex totes
    les coordenades que l'administrador ha introduit a la matriu de
    coordenades de la clase general
588.         public void tancarpanellcoordenades()
589.         {
590.             int i = 0;
591.             clasegenerica.GetInstance().matriucoordenades = new
     double[20, 2];
592.             while (i< (GameObject.Find("teachercanvas/dadesgen/
    paneldrops/ntargg").GetComponent<Dropdown>().value + 1))
593.             {
594.                 int ii = i + 1;
595.                 clasegenerica.GetInstance().matriucoordenades[i
    ,0]= System.Convert.ToDouble(GameObject.Find("/teachercanvas/dad
    esgen/coordenades/panellcoordenades/ScrollView/Viewport/Content/
    latitude" + ii.ToString() + "/Text").GetComponent<Text>().text,
    CultureInfo.InvariantCulture);
596.                 clasegenerica.GetInstance().matriucoordenades[i
    ,1]= System.Convert.ToDouble(GameObject.Find("/teachercanvas/dad
    esgen/coordenades/panellcoordenades/ScrollView/Viewport/Content/
    longitude" + ii.ToString() + "/Text").GetComponent<Text>().text,
    CultureInfo.InvariantCulture);
597.                 Debug.Log(clasegenerica.GetInstance().matriucoo
    rdenades[i, 0]);
598.                 Debug.Log(clasegenerica.GetInstance().matriucoo
    rdenades[i, 1]);
599.                 i = i + 1;
600.             }
601.             panellcoordenades.SetActive(false);
602.         }
603.         //la funció obrirmanualofitxers ens carregarà el panell
    per generar activitats de forma manual o el panell per pujar
    fitxers i imatges directament
604.         // al servidor depenent del botó que premi
605.         public void obrirmodemanualofitxers(int i)
606.         {
607.             if (i == 0)
608.             {
609.                 panelldadesgen.SetActive(true);
610.             }
611.             else
612.             {
613.                 panellfitxers.SetActive(true);
614.             }
615.             escollirmode.SetActive(false);
616.         }
617.     }
```

clasegenerica.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using System;
6. using Vuforia;
7. using UnityEngine.SceneManagement;
8. using UnityEditor;
9. using UnityEngine;
10.      using Vuforia;
11.      using System.IO;
12.      using UnityEngine.EventSystems;
13.      using UnityEngine.SceneManagement;
14.
15.
16.      public class clasegenerica
17.      {
18.          //Aquesta classe es una classe singleton. Només
   s'instancia una vegada i es pot accedir desde qualsevol escena
   sense la necessitat
19.          //de crear un objecte al editor i adjuntar-hi el codi
   com a complement
20.          // ------------------------------------------------
   ------------------------------------------------------------
   ----------------------------
21.          // ------------------------------------------------
   VARIABLES-------------------------------------------------
   ----------------------------
22.          // ------------------------------------------------
   ------------------------------------------------------------
   ----------------------------
23.          public string nomescola; //nom de la escola introduit
   per l'alumne al panell de registre
24.          public List<int> llistaaccions = new List<int>(); //lis
   ta de accions que va disminuint la seva extensió a mesura que
   les proves estiguin llestes
25.          public int[,] ControlAccions; //aquesta matriu de dues
   dimensions serveix per emmagatzemar l'estat de les proves de la
   gimcana
26.          public int ntarg; //número de targets, necesssari per
   carregar la partida desde el perfil de l'alumne.
27.          public int naccions; //número de proves, necesssari per
   carregar la partida desde el perfil de l'alumne.
28.          public int unaototes; //modalitat d'una acció o totes
   (0 o 1), necesssari per carregar la partida desde el perfil de
   l'alumne.
29.          public int ordre; // modalitat d'ordre o sense ordre (0
   o 1) número de targets, necesssari per carregar la partida desde
   el perfil de l'alumne.
30.          public int ntargcreador; //número de targets,
   necesssari per generar la partida desde el perfil de
   l'administrador.
31.          public int naccionscreador; //número de accions,
   necesssari per generar la partida desde el perfil de
   l'administrador.
32.          public int unaototescreador; //modalitat d'una acció o
   totes (0 o 1), necesssari per carregar la partida desde el
   perfil de l'administrador.
33.          public int ordrecreador; //modalitat d'ordre o sense
   ordre (0 o 1), necesssari per carregar la partida desde el
   perfil de l'administrador.
```

```
34.         public string codicreador; //codi de l'activitat
    introduit per l'administrador al moment de crear la gimcana
35.         public string materialaccio; // nom de la imatge
    escollida per l'administrador a l'hora de configurar la textura
    de un cub
36.         public string[] llistaimatges; //lista dels noms de les
    imatges que conformen la prova text+imatges i que
    l'administrador escull a l'hora de crear la prova de text+imatge
37.         public string[] llistarespostes; //lista de les
    respostes que conformen la prova text+imatges i que
    l'administrador introdueix als camps d'entrada del panell
38.         public float longitude; //longitud (GPS) actual de
    l'usuari
39.         public float latitude; //latitud (GPS) actual de
    l'usuari
40.         public double[,] matriucoordenades; // array de dos
    dimensions que conté les coordenades de cada localització
41.         public int numerogrup; //número del grup escollit pels
    alumnes al panell de registre
42.         private static clasegenerica instance; //referencia a
    la classe
43.         // ----------------------------------------------------
    ----------------------------------------------------------------
    ----------------------------
44.         // ----------------------------------------------------
    FUNCIONS-------------------------------------------------------
    ----------------------------
45.         // ----------------------------------------------------
    ----------------------------------------------------------------
    ----------------------------
46.         //el métode GetInstance, instancia una nova clase
    geneèrica en cas que no existeixi ja una, i retorna la que hi ha
    en cas que si.
47.         public static clasegenerica GetInstance()
48.         {
49.             if (instance == null)
50.             {
51.                 instance = new clasegenerica();
52.             }
53.             return instance;
54.         }
55.         //la funció setmatriucontrolaccions serveix per definir
    les dimensions que ha de tenir la matriu controlaccions segons
    el número de targets iv proves
56.         public void setmatriucontrolaccions(int ntargg, int nac
    cionss)
57.         {
58.             ControlAccions = new int[ntargg, naccions];
59.         }
60.         //la corutina start serveix per inicialitzar el servei
    gps al dispositiu de l'usuari quan obre l'aplicació
61.         IEnumerator Start()
62.         {
63.             if (!Input.location.isEnabledByUser) //si la
    localització no està activada al dispositiu de l'usuari fem un
    break
64.             {
65.                 Debug.Log(Input.location.isEnabledByUser);
66.                 yield break;
67.             }
```

```
68.             Input.location.Start(); //inicialitzem el servei de
    localització
69.             while (Input.location.status == LocationServiceStat
    us.Initializing)
70.             {
71.                 yield return new WaitForSeconds(0.5f);
72.                 Debug.Log(Input.location.status);
73.             }
74.             //Una vegada ja tenim accés a les coordenades del
    dispositiu, introduirem els valors latitud i longitud de la
    clase.
75.             latitude = Input.location.lastData.latitude;
76.             latitude = Input.location.lastData.longitude;
77.             yield break;
78.         }
79.         //el métode Update s'utilitza per anar actualitzant a
    cada frame la posició de l'usuari
80.         void Update()
81.         {
82.             latitude = Input.location.lastData.latitude;
83.             longitude = Input.location.lastData.longitude;
84.         }
85.     }
86.
```

cubtextmanager.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using System;
6. using Vuforia;
7. using UnityEngine.SceneManagement;
8. using UnityEngine.EventSystems;
9. using UnityEditor;
10.     using UnityEngine;
11.     using Vuforia;
12.     public class cubtextmanager : MonoBehaviour //Aquesta clase
    gestiona la informació del cub de text i la interacció amb
    l'usuari.
13.     {
14.         // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
15.         // ------------------------------------------------------
    VARIABLES---------------------------------------------------
    ----------------------------
16.         // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
17.         public string inputtascacontructor0;
18.         // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
19.         // ------------------------------------------------------
    FUNCIONS----------------------------------------------------
    ----------------------------
20.         // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
```

```
21.         //el métode OnMouseDown() s'encarrega de gestionar la
   interacció entre l'usuari i el cub. És a dir, comprovarà que
   l'usuari es trobi dins
22.         //l'area de proximitat de la prova i si és així,
   desbloquejarà l'activitat. En cas que l'activitat estigui
   configurada per fer una acció un sol
23.         //cop tsambè comprovarà que no s'hagi fet la prova en
   una altre localització per no repetirla.
24.         void OnMouseDown()
25.         {
26.             if (!EventSystem.current.IsPointerOverGameObject())
27.             {
28.                 //el primer que farem és consultar les
   variables generals de l'activitat.
29.                 int uot = clasegenerica.GetInstance().unaototes
   ; //modalitat d'una acció o totes.
30.                 int ordree = clasegenerica.GetInstance().ordre;
    //modalitat d'ordre o no ordre.
31.                 //després consultem les variables corresponent
   al cub sobre el qual s'acaba de polsar.
32.                 int numerodeaccio = int.Parse(gameObject.name);
    // numero de accio.
33.                 int numerodetarget = int.Parse(gameObject.trans
   form.parent.name); //número de target.
34.                 int boolean = 0; //aquest boleà romandrà en 0
   si es compleixen les condicions per obrir la prova.
35.                 //a continuació es comprovarà si l'usuari es
   troba dins el radi de proximitat de la localització on ha de fer
   la prova
36.                 double earthRadius = 6371;
37.                 double radiantsFactor = System.Math.PI / 180;
38.
39.                 double diffLat = (clasegenerica.GetInstance().m
   atriucoordenades[numerodetarget - 1, 0] - clasegenerica.GetInsta
   nce().latitude) * radiantsFactor;
40.                 double diffLng = (clasegenerica.GetInstance().m
   atriucoordenades[numerodetarget - 1, 1] - clasegenerica.GetInsta
   nce().longitude) * radiantsFactor;
41.                 double a = System.Math.Pow(System.Math.Sin(diff
   Lat / 2), 2) +
42.                                 System.Math.Cos(clasegenerica.GetIn
   stance().latitude * radiantsFactor) *
43.                                 System.Math.Cos(clasegenerica.GetIn
   stance().matriucoordenades[numerodetarget - 1, 0] * radiantsFact
   or) *
44.                                 System.Math.Pow(System.Math.Sin(dif
   fLng / 2), 2);
45.                 double c = 2 * System.Math.Atan2(System.Math.Sq
   rt(a), System.Math.Sqrt(1 - a));
46.                 double res = earthRadius * c;
47.
48.                 double ress = res * 1000; //la variable decimal
   ress ens indica quina es la distancia entre l'usuari i la
   localització establerta pel target
49.                 Debug.Log(ress);
50.                 if (ress >= 5000000) //si la distància supera
   el llindar, s'activarà el panell d'avís amb el missatge "encara
   no has arribar a la localització d'aquesta prova"
51.                 {
```

```
52.                     GameObject.Find("gamemanag").GetComponent<g
   amemanag>().activarpanellavis("encara no has arribat a la
   localització d'aquesta prova");
53.                 }
54.                 else //si la distancia és menor, llavors
   passarem al següent condicional
55.                 {
56.                     if (uot == 0) //si nmomés s'ha de fer una
   acció una sola vegada
57.                     {
58.                         if ((ordree == 0) || (numerodetarget ==
    1)) //si només s'ha de fer una acció una sola vegada i sense
   ordre
59.                         {
60.                             int i = 0;
61.                             while (i < clasegenerica.GetInstanc
   e().ntarg) //es comprovarà que no s'hagi fet l'acció en un altre
   localització
62.                             {
63.                                 if (clasegenerica.GetInstance()
   .ControlAccions[i, numerodeaccio - 1] != 0)
64.                                 {
65.                                     boolean = 1; //si així és,
   el boleà passarà a ser 1.
66.                                 }
67.
68.                                 i = i + 1;
69.                             }
70.                             int j = 0;
71.                             while (j < clasegenerica.GetInstanc
   e().naccions) //es comprovarà que no s'hagi fet alguna acció del
   mateix target
72.                             {
73.                                 if (clasegenerica.GetInstance()
   .ControlAccions[numerodetarget - 1, j] != 0)
74.                                 {
75.                                     boolean = 2; //si així es,
   el boleà passarà a ser 2.
76.                                 }
77.                                 j = j + 1;
78.                             }
79.                             if (boolean == 1) //si el boleà és
   1, s'obrirà el missatge "Acció ja feta en altre localització"
80.                             {
81.                                 GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Acció ja feta en
   altra localització");
82.                             }
83.                             else if (boolean == 2)// si el
   boleà és 2, s'obrirà el missatge "Ja has realitzat una accio
   d'aquest target"
84.                             {
85.                                 GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Ja has realitzat una
   acció d'aquest target");
86.                             }
87.                             // si no és cap de les anteriors,
   llavors continuarà valent 0
88.                         }
```

```
89.                         else // si només s'ha de fer una sola
   acció però de forma ordenada, el que farem serà comprovar que
   les localitzacions anteriors
90.                             //han estat complertes. En el cas
   de la primera no ja que no hi ha localitzacions anteriors, doncs
   és la primera.
91.                         {
92.                             if (numerodeaccio != 1)
93.                             {
94.                                 //a continuació es definirà un
   array que contindrà l'estat de la localització anterior.
95.                                 int[] targetanterior = new int[
   clasegenerica.GetInstance().naccions];
96.                                 int ii = 0;
97.                                 while (ii < clasegenerica.GetIn
   stance().naccions)
98.                                 {
99.                                     targetanterior[ii] = claseg
   enerica.GetInstance().ControlAccions[numerodetarget - 2, ii];
100.                                    ii = ii + 1;
101.                                }
102.                                //si algun dels elements és
   diferent de 0, llavors significa que ja s'ha fet una acció del
   anterior target, i per tant
103.                                //es tornarà a comprovar
   exactament el mateix que en el cas sense ordre: si ja s'ha fet
   aquesta acció en l'anterior target
104.                                //o si j'ha s'ha fet una acció
   del mateix target
105.                                if (Array.Exists(targetanterior
   , Element => Element != 0) == true)
106.                                {
107.                                    int iii = 0;
108.                                    while (iii < clasegenerica.
   GetInstance().ntarg)
109.                                    {
110.                                        if (clasegenerica.GetIn
   stance().ControlAccions[iii, numerodeaccio - 1] != 0)
111.                                        {
112.                                            boolean = 1;
113.                                        }
114.
115.                                        iii = iii + 1;
116.                                    }
117.                                    if (boolean == 1)
118.                                    {
119.                                        GameObject.Find("gamema
   nag").GetComponent<gamemanag>().activarpanellavis("Acció ja feta
   en altra localització");
120.                                    }
121.
122.                                }
123.                                //si tot l'array d'estat està
   format per zeros, llavors significa que encara no s'ha fet cap
   acció de la localització anterior
124.                                //i per tant s'obrirà el
   missatge "Primer has de completar els anteriors targets"
125.                                else
126.                                {
127.                                    boolean = 1;
```

```
128.                                        GameObject.Find("gamemanag"
    ).GetComponent<gamemanag>().activarpanellavis("Primer has de
    completar els anteriors targets");
129.                                    }
130.                                }
131.                            }
132.                        }
133.                        else // si s'han de fer totes les proves
134.                        {
135.                            if (ordree == 1) //i de forma ordenada
136.                            {
137.                                //es comprovarà l'estat de les
    proves de la localització anterior
138.                                int[] targetanterior = new int[clas
    egenerica.GetInstance().naccions];
139.                                int iiii = 0;
140.                                while (iiii < clasegenerica.GetInst
    ance().naccions)
141.                                {
142.                                    targetanterior[iiii] = clasegen
    erica.GetInstance().ControlAccions[numerodetarget - 2, iiii];
143.                                    iiii = iiii + 1;
144.                                }
145.                                //en cas que no s'hagin complert
    totes les proves anteriors, s'obrirà l'avís "Primer has de
    compllletar els anteriors targets"
146.                                if (Array.Exists(targetanterior,
    Element => Element != 0) == false)
147.                                {
148.                                    boolean = 1;
149.                                    GameObject.Find("gamemanag").Ge
    tComponent<gamemanag>().activarpanellavis("Primer has de
    completar els anteriors targets");
150.
151.                                }
152.
153.                            }
154.                        }
155.                        //Finalment es farà una última comprovació:
    Que el mateix cub al qual s'intenta accedir no s'hagi accedit
    anteriorment, per tal de que els 7
156.                        //alumnes no puguin canviar els resultats.
157.                        if ((boolean == 0) && (clasegenerica.GetIns
    tance().ControlAccions[numerodetarget - 1,
    numerodeaccio - 1] == 0))
158.                        {
159.                            GameObject.Find("gamemanag").GetCompone
    nt<gamemanag>().activarpanelltext(inputtascacontructor0,
    numerodetarget, numerodeaccio);
160.                        }
161.                        else if ((clasegenerica.GetInstance().Contr
    olAccions[numerodetarget - 1, numerodeaccio - 1] != 0))
162.                        {
163.                            GameObject.Find("gamemanag").GetCompone
    nt<gamemanag>().activarpanellavis("Ja has realitzat aquesta
    acció");
164.                        }
165.                    }
166.                }
167.            }
168.        }
```

cubtextimatgemanager.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using System;
6. using Vuforia;
7. using UnityEngine.SceneManagement;
8. using UnityEngine.EventSystems;
9. using UnityEditor;
10.      using UnityEngine;
11.      using Vuforia;
12.      public class cubtextimatgemanager : MonoBehaviour //Aquesta
   clase gestiona la informació del cub de text i la interacció amb
   l'usuari.
13.      {
14.          // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
15.          // ------------------------------------------------------
   VARIABLES---------------------------------------------------------
   ----------------------------
16.          // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
17.          public string inputtascacontructor1; // pregunta comú
   per tots els targets
18.          public string imagepath; // nom de la imatge que
   pertoca al corresponent cub
19.          public string resposta; // resposta de la pregunta que
   pertoca al corresponent cub
20.          // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
21.          // ------------------------------------------------------
   FUNCIONS----------------------------------------------------------
   ----------------------------
22.          // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
23.
24.          //el métode OnMouseDown() s'encarrega de gestionar la
   interacció entre l'usuari i el cub. És a dir, comprovarà que
   l'usuari es trobi dins
25.          //l'area de proximitat de la prova i si és així,
   desbloquejarà l'activitat. En cas que l'activitat estigui
   configurada per fer una acció un sol
26.          //cop tsambè comprovarà que no s'hagi fet la prova en
   una altre localització per no repetirla.
27.          void OnMouseDown()
28.          {
29.              if (!EventSystem.current.IsPointerOverGameObject())
30.              {
31.                  //el primer que farem és consultar les
   variables generals de l'activitat.
32.                  int uot = clasegenerica.GetInstance().unaototes
   ; //modalitat d'una acció o totes.
33.                  int ordree = clasegenerica.GetInstance().ordre;
    //modalitat d'ordre o no ordre.
```

```
34.                 //després consultem les variables corresponent
    al cub sobre el qual s'acaba de polsar.
35.                 int numerodeaccio = int.Parse(gameObject.name);
     // numero de accio.
36.                 int numerodetarget = int.Parse(gameObject.trans
    form.parent.name); //número de target.
37.                 int boolean = 0; //aquest boleà romandrà en 0
    si es compleixen les condicions per obrir la prova.
38.                 //a continuació es comprovarà si l'usuari es
    troba dins el radi de proximitat de la localització on ha de fer
    la prova
39.                 double earthRadius = 6371;
40.                 double radiantsFactor = System.Math.PI / 180;
41.
42.                 double diffLat = (clasegenerica.GetInstance().m
    atriucoordenades[numerodetarget - 1, 0] - clasegenerica.GetInsta
    nce().latitude) * radiantsFactor;
43.                 double diffLng = (clasegenerica.GetInstance().m
    atriucoordenades[numerodetarget - 1, 1] - clasegenerica.GetInsta
    nce().longitude) * radiantsFactor;
44.                 double a = System.Math.Pow(System.Math.Sin(diff
    Lat / 2), 2) +
45.                                 System.Math.Cos(clasegenerica.GetIn
    stance().latitude * radiantsFactor) *
46.                                 System.Math.Cos(clasegenerica.GetIn
    stance().matriucoordenades[numerodetarget - 1, 0] * radiantsFact
    or) *
47.                                 System.Math.Pow(System.Math.Sin(dif
    fLng / 2), 2);
48.                 double c = 2 * System.Math.Atan2(System.Math.Sq
    rt(a), System.Math.Sqrt(1 - a));
49.                 double res = earthRadius * c;
50.
51.                 double ress = res * 1000; //la variable decimal
    ress ens indica quina es la distancia entre l'usuari i la
    localització establerta pel target
52.                 Debug.Log(ress);
53.                 if (ress>=5000000) //si la distància supera el
    llindar, s'activarà el panell d'avís amb el missatge "encara no
    has arribar a la localització d'aquesta prova"
54.                 {
55.                     GameObject.Find("gamemanag").GetComponent<g
    amemanag>().activarpanellavis("encara no has arribat a la
    localització d'aquesta prova");
56.                 }
57.                 else //si la distancia és menor, llavors
    passarem al següent condicional
58.                 {
59.                     if (uot==0) //si nmomés s'ha de fer una
    acció una sola vegada
60.                     {
61.                         if((ordree==0)||(numerodetarget==1)) //
    si només s'ha de fer una acció una sola vegada i sense ordre
62.                         {
63.                             int i = 0;
64.                             while (i<clasegenerica.GetInstance(
    ).ntarg) //es comprovarà que no s'hagi fet l'acció en un altre
    localització
65.                             {
66.                                 if (clasegenerica.GetInstance()
    .ControlAccions[i,numerodeaccio-1]!=0)
```

```
67.                                     {
68.                                         boolean = 1; //si així és,
   el boleà passarà a ser 1.
69.                                     }
70.
71.                                     i = i + 1;
72.                                 }
73.                                 int j = 0;
74.                                 while (j < clasegenerica.GetInstanc
   e().naccions) //es comprovarà que no s'hagi fet alguna acció del
   mateix target
75.                                 {
76.                                     if (clasegenerica.GetInstance()
   .ControlAccions[numerodetarget-1,j] != 0)
77.                                     {
78.                                         boolean = 2; //si així es,
   el boleà passarà a ser 2.
79.                                     }
80.                                     j = j + 1;
81.                                 }
82.                                 if (boolean == 1) //si el boleà és
   1, s'obrirà el missatge "Acció ja feta en altre localització"
83.                                 {
84.                                     GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Acció ja feta en
   altra localització");
85.                                 }
86.                                 else if (boolean == 2)// si el
   boleà és 2, s'obrirà el missatge "Ja has realitzat una accio
   d'aquest target"
87.                                 {
88.                                     GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Ja has realitzat una
   acció d'aquest target");
89.                                 }
90.                                 // si no és cap de les anteriors,
   llavors continuarà valent 0
91.                             }
92.                             else // si només s'ha de fer una sola
   acció però de forma ordenada, el que farem serà comprovar que
   les localitzacions anteriors
93.                                  //han estat complertes. En el cas
   de la primera no ja que no hi ha localitzacions anteriors, doncs
   és la primera.
94.                             {
95.                                 if(numerodeaccio!=1)
96.                                 {
97.                                     //a continuació es definirà un
   array que contindrà l'estat de la localització anterior.
98.                                     int[] targetanterior = new int[
   clasegenerica.GetInstance().naccions];
99.                                     int ii = 0;
100.                                    while (ii < clasegenerica.GetIn
   stance().naccions)
101.                                    {
102.                                        targetanterior[ii] = claseg
   enerica.GetInstance().ControlAccions[numerodetarget - 2, ii];
103.                                        ii = ii + 1;
104.                                    }
```

```
105.                                    //si algun dels elements és
   diferent de 0, llavors significa que ja s'ha fet una acció del
   anterior target, i per tant
106.                                    //es tornarà a comprovar
   exactament el mateix que en el cas sense ordre: si ja s'ha fet
   aquesta acció en l'anterior target
107.                                    //o si j'ha s'ha fet una acció
   del mateix target
108.                                    if (Array.Exists(targetanterior
   ,Element => Element!=0)==true)
109.                                    {
110.                                        int iii = 0;
111.                                        while (iii < clasegenerica.
   GetInstance().ntarg)
112.                                        {
113.                                            if (clasegenerica.GetIn
   stance().ControlAccions[iii, numerodeaccio - 1] != 0)
114.                                            {
115.                                                boolean = 1;
116.                                            }
117.
118.                                            iii = iii + 1;
119.                                        }
120.                                        if (boolean == 1)
121.                                        {
122.                                            GameObject.Find("gamema
   nag").GetComponent<gamemanag>().activarpanellavis("Acció ja feta
   en altra localització");
123.                                        }
124.
125.                                    }
126.                                    //si tot l'array d'estat està
   format per zeros, llavors significa que encara no s'ha fet cap
   acció de la localització anterior
127.                                    //i per tant s'obrirà el
   missatge "Primer has de completar els anteriors targets"
128.                                    else
129.                                    {
130.                                        boolean = 1;
131.                                        GameObject.Find("gamemanag"
   ).GetComponent<gamemanag>().activarpanellavis("Primer has de
   completar els anteriors targets");
132.                                    }
133.                                }
134.                            }
135.                        }
136.                        else // si s'han de fer totes les proves
137.                        {
138.                            if (ordree==1) //i de forma ordenada
139.                            {
140.                                //es comprovarà l'estat de les
   proves de la localització anterior
141.                                int[] targetanterior = new int[clas
   egenerica.GetInstance().naccions];
142.                                int iiii = 0;
143.                                while (iiii < clasegenerica.GetInst
   ance().naccions)
144.                                {
145.                                    targetanterior[iiii] = clasegen
   erica.GetInstance().ControlAccions[numerodetarget - 2, iiii];
146.                                    iiii = iiii + 1;
```

```
147.                                }
148.                                //en cas que no s'hagin complert
   totes les proves anteriors, s'obrirà l'avís "Primer has de
   compllletar els anteriors targets"
149.                                if (Array.Exists(targetanterior,
   Element => Element != 0) == false)
150.                                {
151.                                    boolean = 1;
152.                                    GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Primer has de
   completar els anteriors targets");
153.
154.                                }
155.
156.                            }
157.                        }
158.                        //Finalment es farà una última comprovació:
   Que el mateix cub al qual s'intenta accedir no s'hagi accedit
   anteriorment, per tal de que els 7
159.                        //alumnes no puguin canviar els resultats.
160.                        if ((boolean==0)&&(clasegenerica.GetInstanc
   e().ControlAccions[numerodetarget-1,numerodeaccio-1]==0))
161.                        {
162.                            GameObject.Find("gamemanag").GetCompone
   nt<gamemanag>().activarpanelltextimatge(inputtascacontructor1,
   imagepath, resposta, numerodetarget, numerodeaccio);
163.                        }
164.                        else if ((boolean == 0) && (clasegenerica.G
   etInstance().ControlAccions[numerodetarget - 1,
   numerodeaccio - 1] !=0))
165.                        {
166.                            GameObject.Find("gamemanag").GetCompone
   nt<gamemanag>().activarpanellavis("Ja has realitzat aquesta
   acció");
167.                        }
168.                    }
169.                }
170.            }
171.        }
```

formularimanager.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using System;
6. using Vuforia;
7. using UnityEngine.SceneManagement;
8. using UnityEngine.EventSystems;
9. using UnityEditor;
10.      using UnityEngine;
11.      using Vuforia;
12.
13.      public class formularimanager : MonoBehaviour //Aquesta
   clase gestiona la informació del cub de formulari i la
   interacció amb l'usuari.
14.      {
15.          // ------------------------------------------------------
   ------------------------------------------------------------------
   ----------------------------
```

```
16.         // ------------------------------------------------------
    VARIABLES-----------------------------------------------------
    ----------------------------
17.         // ------------------------------------------------------
    ------------------------------------------------------------------
    ----------------------------
18.         public string linkdelformulari;
19.         // ------------------------------------------------------
    ------------------------------------------------------------------
    ----------------------------
20.         // ------------------------------------------------------
    FUNCIONS------------------------------------------------------
    ----------------------------
21.         // ------------------------------------------------------
    ------------------------------------------------------------------
    ----------------------------
22.         //el métode OnMouseDown() s'encarrega de gestionar la
    interacció entre l'usuari i el cub. És a dir, comprovarà que
    l'usuari es trobi dins
23.         //l'area de proximitat de la prova i si és així,
    desbloquejarà l'activitat. En cas que l'activitat estigui
    configurada per fer una acció un sol
24.         //cop tsambè comprovarà que no s'hagi fet la prova en
    una altre localització per no repetirla.
25.         void OnMouseDown()
26.         {
27.             if (!EventSystem.current.IsPointerOverGameObject())
28.             {
29.                 //el primer que farem és consultar les
    variables generals de l'activitat.
30.                 int uot = clasegenerica.GetInstance().unaototes
    ; //modalitat d'una acció o totes.
31.                 int ordree = clasegenerica.GetInstance().ordre;
     //modalitat d'ordre o no ordre.
32.                 //després consultem les variables corresponent
    al cub sobre el qual s'acaba de polsar.
33.                 int numerodeaccio = int.Parse(gameObject.name);
     // numero de accio.
34.                 int numerodetarget = int.Parse(gameObject.trans
    form.parent.name); //número de target.
35.                 int boolean = 0; //aquest boleà romandrà en 0
    si es compleixen les condicions per obrir la prova.
36.                 //a continuació es comprovarà si l'usuari es
    troba dins el radi de proximitat de la localització on ha de fer
    la prova
37.                 double earthRadius = 6371;
38.                 double radiantsFactor = System.Math.PI / 180;
39.
40.                 double diffLat = (clasegenerica.GetInstance().m
    atriucoordenades[numerodetarget - 1, 0] - clasegenerica.GetInsta
    nce().latitude) * radiantsFactor;
41.                 double diffLng = (clasegenerica.GetInstance().m
    atriucoordenades[numerodetarget - 1, 1] - clasegenerica.GetInsta
    nce().longitude) * radiantsFactor;
42.                 double a = System.Math.Pow(System.Math.Sin(diff
    Lat / 2), 2) +
43.                                 System.Math.Cos(clasegenerica.GetIn
    stance().latitude * radiantsFactor) *
44.                                 System.Math.Cos(clasegenerica.GetIn
    stance().matriucoordenades[numerodetarget - 1, 0] * radiantsFact
    or) *
```

```
45.                                 System.Math.Pow(System.Math.Sin(dif
   fLng / 2), 2);
46.                     double c = 2 * System.Math.Atan2(System.Math.Sq
   rt(a), System.Math.Sqrt(1 - a));
47.                     double res = earthRadius * c;
48.
49.                     double ress = res * 1000; //la variable decimal
   ress ens indica quina es la distancia entre l'usuari i la
   localització establerta pel target
50.                     Debug.Log(ress);
51.                     if (ress >= 5000000) //si la distància supera
   el llindar, s'activarà el panell d'avís amb el missatge "encara
   no has arribar a la localització d'aquesta prova"
52.                     {
53.                         GameObject.Find("gamemanag").GetComponent<g
   amemanag>().activarpanellavis("encara no has arribat a la
   localització d'aquesta prova");
54.                     }
55.                     else //si la distancia és menor, llavors
   passarem al següent condicional
56.                     {
57.                         if (uot == 0) //si nmomés s'ha de fer una
   acció una sola vegada
58.                         {
59.                             if ((ordree == 0) || (numerodetarget ==
    1)) //si només s'ha de fer una acció una sola vegada i sense
   ordre
60.                             {
61.                                 int i = 0;
62.                                 while (i < clasegenerica.GetInstanc
   e().ntarg) //es comprovarà que no s'hagi fet l'acció en un altre
   localització
63.                                 {
64.                                     if (clasegenerica.GetInstance()
   .ControlAccions[i, numerodeaccio - 1] != 0)
65.                                     {
66.                                         boolean = 1; //si així és,
   el boleà passarà a ser 1.
67.                                     }
68.
69.                                     i = i + 1;
70.                                 }
71.                                 int j = 0;
72.                                 while (j < clasegenerica.GetInstanc
   e().naccions) //es comprovarà que no s'hagi fet alguna acció del
   mateix target
73.                                 {
74.                                     if (clasegenerica.GetInstance()
   .ControlAccions[numerodetarget - 1, j] != 0)
75.                                     {
76.                                         boolean = 2; //si així es,
   el boleà passarà a ser 2.
77.                                     }
78.                                     j = j + 1;
79.                                 }
80.                                 if (boolean == 1) //si el boleà és
   1, s'obrirà el missatge "Acció ja feta en altre localització"
81.                                 {
82.                                     GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Acció ja feta en
   altra localització");
```

```
83.                                 }
84.                                 else if (boolean == 2)// si el
   boleà és 2, s'obrirà el missatge "Ja has realitzat una accio
   d'aquest target"
85.                                 {
86.                                     GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Ja has realitzat una
   acció d'aquest target");
87.                                 }
88.                                 // si no és cap de les anteriors,
   llavors continuarà valent 0
89.                             }
90.                             else // si només s'ha de fer una sola
   acció però de forma ordenada, el que farem serà comprovar que
   les localitzacions anteriors
91.                                  //han estat complertes. En el cas
   de la primera no ja que no hi ha localitzacions anteriors, doncs
   és la primera.
92.                             {
93.                                 if (numerodeaccio != 1)
94.                                 {
95.                                     //a continuació es definirà un
   array que contindrà l'estat de la localització anterior.
96.                                     int[] targetanterior = new int[
   clasegenerica.GetInstance().naccions];
97.                                     int ii = 0;
98.                                     while (ii < clasegenerica.GetIn
   stance().naccions)
99.                                     {
100.                                        targetanterior[ii] = claseg
   enerica.GetInstance().ControlAccions[numerodetarget - 2, ii];
101.                                        ii = ii + 1;
102.                                    }
103.                                    //si algun dels elements és
   diferent de 0, llavors significa que ja s'ha fet una acció del
   anterior target, i per tant
104.                                    //es tornarà a comprovar
   exactament el mateix que en el cas sense ordre: si ja s'ha fet
   aquesta acció en l'anterior target
105.                                    //o si j'ha s'ha fet una acció
   del mateix target
106.                                    if (Array.Exists(targetanterior
   , Element => Element != 0) == true)
107.                                    {
108.                                        int iii = 0;
109.                                        while (iii < clasegenerica.
   GetInstance().ntarg)
110.                                        {
111.                                            if (clasegenerica.GetIn
   stance().ControlAccions[iii, numerodeaccio - 1] != 0)
112.                                            {
113.                                                boolean = 1;
114.                                            }
115.
116.                                            iii = iii + 1;
117.                                        }
118.                                        if (boolean == 1)
119.                                        {
120.                                            GameObject.Find("gamema
   nag").GetComponent<gamemanag>().activarpanellavis("Acció ja feta
   en altra localització");
```

```
121.                                     }
122.
123.                                 }
124.                                 //si tot l'array d'estat està
    format per zeros, llavors significa que encara no s'ha fet cap
    acció de la localització anterior
125.                                 //i per tant s'obrirà el
    missatge "Primer has de completar els anteriors targets"
126.                                 else
127.                                 {
128.                                     boolean = 1;
129.                                     GameObject.Find("gamemanag"
    ).GetComponent<gamemanag>().activarpanellavis("Primer has de
    completar els anteriors targets");
130.                                 }
131.                             }
132.                         }
133.                     }
134.                     else // si s'han de fer totes les proves
135.                     {
136.                         if (ordree == 1) //i de forma ordenada
137.                         {
138.                             //es comprovarà l'estat de les
    proves de la localització anterior
139.                             int[] targetanterior = new int[clas
    egenerica.GetInstance().naccions];
140.                             int iiii = 0;
141.                             while (iiii < clasegenerica.GetInst
    ance().naccions)
142.                             {
143.                                 targetanterior[iiii] = clasegen
    erica.GetInstance().ControlAccions[numerodetarget - 2, iiii];
144.                                 iiii = iiii + 1;
145.                             }
146.                             //en cas que no s'hagin complert
    totes les proves anteriors, s'obrirà l'avís "Primer has de
    compllletar els anteriors targets"
147.                             if (Array.Exists(targetanterior,
    Element => Element != 0) == false)
148.                             {
149.                                 boolean = 1;
150.                                 GameObject.Find("gamemanag").Ge
    tComponent<gamemanag>().activarpanellavis("Primer has de
    completar els anteriors targets");
151.
152.                             }
153.
154.                         }
155.                     }
156.                     //Finalment es farà una última comprovació:
    Que el mateix cub al qual s'intenta accedir no s'hagi accedit
    anteriorment, per tal de que els 7
157.                     //alumnes no puguin canviar els resultats.
158.                     if ((boolean == 0) && (clasegenerica.GetIns
    tance().ControlAccions[numerodetarget - 1,
    numerodeaccio - 1] == 0))
159.                     {
160.                         GameObject.Find("gamemanag").GetCompone
    nt<gamemanag>().activarformulari(linkdelformulari,
    numerodetarget, numerodeaccio);
161.                     }
```

```
                    else if ((boolean == 0) && (clasegenerica.G
    etInstance().ControlAccions[numerodetarget - 1,
    numerodeaccio - 1] != 0))
                    {
                        GameObject.Find("gamemanag").GetCompone
    nt<gamemanag>().activarpanellavis("Ja has realitzat aquesta
    acció");
                    }
                }
            }
        }
    }
```

trivialmanager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using Vuforia;
using UnityEngine.SceneManagement;
using UnityEngine.EventSystems;
using UnityEditor;
    using UnityEngine;
    using Vuforia;
    public class trivialmanager : MonoBehaviour //Aquesta clase
    gestiona la informació del cub de trivial i la interacció amb
    l'usuari.
    {
        // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
        // ------------------------------------------------------
    VARIABLES---------------------------------------------------
    ----------------------------
        // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
        public string pregunta; //pregunta corresponent al cub
        public string resposta1; //resposta a corresponent al
    cub
        public string resposta2; //resposta b corresponent al
    cub
        public string resposta3; //resposta c corresponent al
    cub
        public string resposta4; //resposta d corresponent al
    cub
        public string respostabona; //resposta correcta
    corresponent al cub
        // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
        // ------------------------------------------------------
    FUNCIONS----------------------------------------------------
    ----------------------------
        // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
```

```
26.         //el métode OnMouseDown() s'encarrega de gestionar la
   interacció entre l'usuari i el cub. És a dir, comprovarà que
   l'usuari es trobi dins
27.         //l'area de proximitat de la prova i si és així,
   desbloquejarà l'activitat. En cas que l'activitat estigui
   configurada per fer una acció un sol
28.         //cop tsambè comprovarà que no s'hagi fet la prova en
   una altre localització per no repetirla.
29.         void OnMouseDown()
30.         {
31.             if (!EventSystem.current.IsPointerOverGameObject())
32.             {
33.                 //el primer que farem és consultar les
   variables generals de l'activitat.
34.                 int uot = clasegenerica.GetInstance().unaototes
   ; //modalitat d'una acció o totes.
35.                 int ordree = clasegenerica.GetInstance().ordre;
    //modalitat d'ordre o no ordre.
36.                 //després consultem les variables corresponent
   al cub sobre el qual s'acaba de polsar.
37.                 int numerodeaccio = int.Parse(gameObject.name);
    // numero de accio.
38.                 int numerodetarget = int.Parse(gameObject.trans
   form.parent.name); //número de target.
39.                 int boolean = 0; //aquest boleà romandrà en 0
   si es compleixen les condicions per obrir la prova.
40.                 //a continuació es comprovarà si l'usuari es
   troba dins el radi de proximitat de la localització on ha de fer
   la prova
41.                 double earthRadius = 6371;
42.                 double radiantsFactor = System.Math.PI / 180;
43.
44.                 double diffLat = (clasegenerica.GetInstance().m
   atriucoordenades[numerodetarget - 1, 0] - clasegenerica.GetInsta
   nce().latitude) * radiantsFactor;
45.                 double diffLng = (clasegenerica.GetInstance().m
   atriucoordenades[numerodetarget - 1, 1] - clasegenerica.GetInsta
   nce().longitude) * radiantsFactor;
46.                 double a = System.Math.Pow(System.Math.Sin(diff
   Lat / 2), 2) +
47.                                 System.Math.Cos(clasegenerica.GetIn
   stance().latitude * radiantsFactor) *
48.                                 System.Math.Cos(clasegenerica.GetIn
   stance().matriucoordenades[numerodetarget - 1, 0] * radiantsFact
   or) *
49.                                 System.Math.Pow(System.Math.Sin(dif
   fLng / 2), 2);
50.                 double c = 2 * System.Math.Atan2(System.Math.Sq
   rt(a), System.Math.Sqrt(1 - a));
51.                 double res = earthRadius * c;
52.
53.                 double ress = res * 1000; //la variable decimal
   ress ens indica quina es la distancia entre l'usuari i la
   localització establerta pel target
54.                 Debug.Log(ress);
55.                 if (ress >= 5000000) //si la distància supera
   el llindar, s'activarà el panell d'avís amb el missatge "encara
   no has arribar a la localització d'aquesta prova"
56.                 {
```

```
57.                     GameObject.Find("gamemanag").GetComponent<g
   amemanag>().activarpanellavis("encara no has arribat a la
   localització d'aquesta prova");
58.                 }
59.                 else //si la distancia és menor, llavors
   passarem al següent condicional
60.                 {
61.                     if (uot == 0) //si nmomés s'ha de fer una
   acció una sola vegada
62.                     {
63.                         if ((ordree == 0) || (numerodetarget ==
    1)) //si només s'ha de fer una acció una sola vegada i sense
   ordre
64.                         {
65.                             int i = 0;
66.                             while (i < clasegenerica.GetInstanc
   e().ntarg) //es comprovarà que no s'hagi fet l'acció en un altre
   localització
67.                             {
68.                                 if (clasegenerica.GetInstance()
   .ControlAccions[i, numerodeaccio - 1] != 0)
69.                                 {
70.                                     boolean = 1; //si així és,
   el boleà passarà a ser 1.
71.                                 }
72.
73.                                 i = i + 1;
74.                             }
75.                             int j = 0;
76.                             while (j < clasegenerica.GetInstanc
   e().naccions) //es comprovarà que no s'hagi fet alguna acció del
   mateix target
77.                             {
78.                                 if (clasegenerica.GetInstance()
   .ControlAccions[numerodetarget - 1, j] != 0)
79.                                 {
80.                                     boolean = 2; //si així es,
   el boleà passarà a ser 2.
81.                                 }
82.                                 j = j + 1;
83.                             }
84.                             if (boolean == 1) //si el boleà és
   1, s'obrirà el missatge "Acció ja feta en altre localització"
85.                             {
86.                                 GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Acció ja feta en
   altra localització");
87.                             }
88.                             else if (boolean == 2)// si el
   boleà és 2, s'obrirà el missatge "Ja has realitzat una accio
   d'aquest target"
89.                             {
90.                                 GameObject.Find("gamemanag").Ge
   tComponent<gamemanag>().activarpanellavis("Ja has realitzat una
   acció d'aquest target");
91.                             }
92.                             // si no és cap de les anteriors,
   llavors continuarà valent 0
93.                         }
```

```
94.                         else // si només s'ha de fer una sola
   acció però de forma ordenada, el que farem serà comprovar que
   les localitzacions anteriors
95.                             //han estat complertes. En el cas
   de la primera no ja que no hi ha localitzacions anteriors, doncs
   és la primera.
96.                         {
97.                             if (numerodeaccio != 1)
98.                             {
99.                                 //a continuació es definirà un
   array que contindrà l'estat de la localització anterior.
100.                                int[] targetanterior = new int[
   clasegenerica.GetInstance().naccions];
101.                                int ii = 0;
102.                                while (ii < clasegenerica.GetIn
   stance().naccions)
103.                                {
104.                                    targetanterior[ii] = claseg
   enerica.GetInstance().ControlAccions[numerodetarget - 2, ii];
105.                                    ii = ii + 1;
106.                                }
107.                                //si algun dels elements és
   diferent de 0, llavors significa que ja s'ha fet una acció del
   anterior target, i per tant
108.                                //es tornarà a comprovar
   exactament el mateix que en el cas sense ordre: si ja s'ha fet
   aquesta acció en l'anterior target
109.                                //o si j'ha s'ha fet una acció
   del mateix target
110.                                if (Array.Exists(targetanterior
   , Element => Element != 0) == true)
111.                                {
112.                                    int iii = 0;
113.                                    while (iii < clasegenerica.
   GetInstance().ntarg)
114.                                    {
115.                                        if (clasegenerica.GetIn
   stance().ControlAccions[iii, numerodeaccio - 1] != 0)
116.                                        {
117.                                            boolean = 1;
118.                                        }
119.
120.                                        iii = iii + 1;
121.                                    }
122.                                    if (boolean == 1)
123.                                    {
124.                                        GameObject.Find("gamema
   nag").GetComponent<gamemanag>().activarpanellavis("Acció ja feta
   en altra localització");
125.                                    }
126.
127.                                }
128.                                //si tot l'array d'estat està
   format per zeros, llavors significa que encara no s'ha fet cap
   acció de la localització anterior
129.                                //i per tant s'obrirà el
   missatge "Primer has de completar els anteriors targets"
130.                                else
131.                                {
132.                                    boolean = 1;
```

```
133.                                    GameObject.Find("gamemanag"
    ).GetComponent<gamemanag>().activarpanellavis("Primer has de
    completar els anteriors targets");
134.                                }
135.                            }
136.                        }
137.                    }
138.                    else // si s'han de fer totes les proves
139.                    {
140.                        if (ordree == 1) //i de forma ordenada
141.                        {
142.                            //es comprovarà l'estat de les
    proves de la localització anterior
143.                            int[] targetanterior = new int[clas
    egenerica.GetInstance().naccions];
144.                            int iiii = 0;
145.                            while (iiii < clasegenerica.GetInst
    ance().naccions)
146.                            {
147.                                targetanterior[iiii] = clasegen
    erica.GetInstance().ControlAccions[numerodetarget - 2, iiii];
148.                                iiii = iiii + 1;
149.                            }
150.                            //en cas que no s'hagin complert
    totes les proves anteriors, s'obrirà l'avís "Primer has de
    compllletar els anteriors targets"
151.                            if (Array.Exists(targetanterior,
    Element => Element != 0) == false)
152.                            {
153.                                boolean = 1;
154.                                GameObject.Find("gamemanag").Ge
    tComponent<gamemanag>().activarpanellavis("Primer has de
    completar els anteriors targets");
155.
156.                            }
157.
158.                        }
159.                    }
160.                    //Finalment es farà una última comprovació:
    Que el mateix cub al qual s'intenta accedir no s'hagi accedit
    anteriorment, per tal de que els 7
161.                    //alumnes no puguin canviar els resultats.
162.                    if ((boolean == 0) && (clasegenerica.GetIns
    tance().ControlAccions[numerodetarget - 1,
    numerodeaccio - 1] == 0))
163.                    {
164.                        StartCoroutine(activareltrivial());
165.                    }
166.                    else if ((boolean == 0) && (clasegenerica.G
    etInstance().ControlAccions[numerodetarget - 1,
    numerodeaccio - 1] != 0))
167.                    {
168.                        GameObject.Find("gamemanag").GetCompone
    nt<gamemanag>().activarpanellavis("Ja has realitzat aquesta
    acció");
169.                    }
170.                }
171.            }
172.        }
173.        // la corrutina activareltrivial s'encarrega d'activar
    el panell del trivial amb la informació continguda al cub
```

```
174.         // la rao per la que utilitzo una corutina es per que
   abans d'obrir el panell, esperarem un segon ja que pel contrari,
   si obrim el cub desde un lloc
175.         //de la pantalla que coincideixi amb alguna de les
   opcions del trivial, aquesta seria pulsada immediatament.
176.         public IEnumerator activareltrivial()
177.         {
178.             yield return new WaitForSeconds(1);
179.             int uot = clasegenerica.GetInstance().unaototes;
180.             int ordree = clasegenerica.GetInstance().ordre;
181.             int numerodeaccio = int.Parse(gameObject.name);
182.             int numerodetarget = int.Parse(gameObject.transform
   .parent.name);
183.             GameObject.Find("gamemanag").GetComponent<gamemanag
   >().activarpanelltrivial(pregunta, resposta1, resposta2,
   resposta3, resposta4, respostabona, numerodetarget,
   numerodeaccio);
184.         }
185.     }
```

treurefills.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class treurefills : MonoBehaviour
6. {
7.     // ------------------------------------------------------
   ------------------------------------------------------------
   ------------------------
8.     // ------------------------------------------------------
   FUNCIONS-----------------------------------------------------
   ----------------------------
9.     // ------------------------------------------------------
   ------------------------------------------------------------
   ------------------------
10.         // la funció treurefills() s'encarrega d'eliminar tot
   el contingut de la scrollview del panell de resultats cada
   vegada que tanquem el panell7
11.         // de puntuacions. Es tornarà a omplir amb la funció
   setgridd() del gamemang quan l'alumne premi el botó mostrar
   resultats
12.         public void treurefils()
13.         {
14.             foreach (Transform child in transform)
15.             {
16.                 Destroy(child.gameObject);
17.             }
18.         }
19.     }
```

fabricaa.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class treurefills : MonoBehaviour
```

```
{
    // ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------
    // ------------------------------------------------
    FUNCIONS-------------------------------------------------------
    ----------------------------
    // ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------
        // la funció treurefills() s'encarrega d'eliminar tot
    el contingut de la scrollview del panell de resultats cada
    vegada que tanquem el panell7
        // de puntuacions. Es tornarà a omplir amb la funció
    setgridd() del gamemang quan l'alumne premi el botó mostrar
    resultats
        public void treurefils()
        {
            foreach (Transform child in transform)
            {
                Destroy(child.gameObject);
            }
        }
    }
```

loadscene.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class loadscene : MonoBehaviour //aquesta classe serveix
per carregar les diferents escenes del joc
{
    // ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------
    // ------------------------------------------------
    FUNCIONS-------------------------------------------------------
    ----------------------------
        // ------------------------------------------------------
    ------------------------------------------------------------
    ----------------------------
        //la funció loadscene() rep com a entrada un enter. Si
    és un 0, carregarà la escena de l'administrador, si és un 1, la
    de l'alumne
        //si es un 2, carregarà el menú principal on s'escull
    entre el perfil d'administrador o d'estudiant.
        public void loadscenee(int numer)
        {
            if (numer==1)
            {
                SceneManager.LoadScene("studentscene");
            }
            else if (numer==2)
            {
                SceneManager.LoadScene("SampleScene");
            }
```

```
23.                 else
24.                 {
25.                     SceneManager.LoadScene("teacherscene");
26.
27.                 }
28.             }
29.         }
30.         <?php
31.         $server = "localhost";
32.         $username = "root";
33.         $password = "";
34.         $db = "resultats";
35.         $connect = new mysqli ($server,$username,$password,$db);
36.         ?>
```

userInsert.php

```
1. <?php
2. include('connection.php');
3.
4. $escola = $_POST['addescola'];
5. $grup = $_POST['addgrup'];
6. $puntuacio = $_POST['addpuntuacio'];
7.
8. $sql = "insert into usuaris (escola, grup, puntuacio) values
   ('".$escola"','".$grup"','".$puntuacio"')";
9. $result = mysqli_query($connect,$sql);
10.
11.         ?>
```

connection.cs

```
1. <?php
2. $allowedExts = array("txt");
3. $thefile = $_FILES['file'];
4. $target_path = "ficheros/";
5. $target_path = $target_path.basename($_FILES['file']['name']);
6.
7. if (move_uploaded_file($_FILES['file']['tmp_name'],$target_path)
   )
8. {
9.           echo "El archivo se ha subido con exito";
10.         }
11.         else
12.         {
13.                 echo "ha ocurrido un error, revisa el codigo";
14.         }
15.
16.         ?>
```

cargararchivo.php

```
17.         <?php
18.         $allowedExts = array("jpg", "jpeg", "gif", "png");
19.         $thefile = $_FILES['file'];
20.         $target_path = "imagenes/";
```

```
21.       $target_path = $target_path.basename($_FILES['file']['name'
   ]);
22.
23.       if (move_uploaded_file($_FILES['file']['tmp_name'],$target_
   path))
24.       {
25.                 echo "El archivo se ha subido con exito";
26.       }
27.       else
28.       {
29.                 echo "ha ocurrido un error, revisa el codigo";
30.       }
31.
32.       ?>
```

cargararchivotexto.php

```
1. <?php
2. $allowedExts = array("txt");
3. $thefile = $_FILES['file'];
4. $target_path = "ficheros/";
5. $target_path = $target_path.basename($_FILES['file']['name']);
6.
7. if (move_uploaded_file($_FILES['file']['tmp_name'],$target_path)
   )
8. {
9.         echo "El archivo se ha subido con exito";
10.       }
11.       else
12.       {
13.                 echo "ha ocurrido un error, revisa el codigo";
14.       }
15.
16.       ?>
```